VŠB – Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Analýza sentimentů v textech

Sentiment Analysis in Text

2020 Marek Hajdík

VŠB - Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Zadání diplomové práce

Student: **Bc. Marek Hajdík**

Studijní program: N2647 Informační a komunikační technologie

Studijní obor: 2612T025 Informatika a výpočetní technika

Téma: Analýza sentimentů v textech
Sentiment Analysis in Text

Jazyk vypracování: čeština

Zásady pro vypracování:

Cílem práce je popsat metody pro analýzu sentimentu v textových dokumentů. Vybrané metody budou popsány, implementovány a ověřeny na vhodných datových sadách.

Práce bude obsahovat:
1. Popis metod pro analýzu sentimentu v textech.
2. Popis vybraných algoritmů.
3. Návrh implementace a implementace algoritmů.
4. Experimentální ověření algoritmů a jejich porovnání.

Seznam doporučené odborné literatury:

[1] Chris Manning and Hinrich Schütze, Foundations of Statistical Natural Language Processing, MIT Press. Cambridge, MA: May 1999
[2] Dan Jurafsky: Speech and Language Processing, Pearson Education, 2014
[3] Bing Liu: Sentiment Analysis: Mining Opinions, Sentiments, and Emotions, Cambridge University Press; 1 edition (June 4, 2015)
[4] Charu C. Aggarwal, Machine learning for text. New York, NY: Springer Science+Business Media, 2018. ISBN 978-3-319-73530-6.

Formální náležitosti a rozsah diplomové práce stanoví pokyny pro vypracování zveřejněné na webových stránkách fakulty.

Vedoucí diplomové práce: **doc. Ing. Jan Platoš, Ph.D.**

Datum zadání: 01.09.2019
Datum odevzdání: 30.04.2020

doc. Ing. Jan Platoš, Ph.D.
vedoucí katedry

prof. Ing. Pavel Brandštetter, CSc.
děkan fakulty

Prohlašuji, že jsem tuto diplomovou práci vypracoval samostatně. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

V Ostravě 14. května 2020 ..... Hajdek .....

Rád bych na tomto místě poděkoval vedoucímu diplomové práce doc. Ing. Janu Platošovi, Ph.D. za jeho odborné rady, konzultace a vstřícný přistup při tvorbě této práce. Také bych chtěl poděkovat své rodině, která mě po celou dobu podporovala.

Abstrakt

Tato diplomová práce se zabývá analýzou sentimentu v textových dokumentech. Cílem bylo popsat vybrané metody, implementovat je a otestovat na vhodných datových sadách. Teoretická část se věnuje problematice zpracování přirozeného jazyka a metodám pro rozpoznání sentimentu. Jsou zde uvedeny úrovně analýzy, příprava textu, možné komplikace a rozdělení metod pro řešení tohoto problému. Dále je stručný popis metod, které v současnosti dosahují nejlepších výsledků při klasifikaci sentimentu a podrobný popis vybraných algoritmů, použitých v praktické části práce. Všechny vybrané algoritmy se řádí do oblasti strojového učení a byly vybrány na základě aktuálních trendů. Praktická část se věnuje detailům implementace vybraných metod a návrhům neuronových sítí. Poté jsou všechny metody ověřeny a porovnány pomocí různých experimentů, jež testují účinek předzpracování textu, optimalizaci hyperparametrů a vliv rozšíření struktur neuronových sítí. Pro provedení experimentů byly vybrány tři datové sady tak, aby ověřily metody při nedostatečném a nadměrném množství dat.

Klíčová slova: analýza sentimentu; klasifikace textů; strojové učení; zpracování přirozeného jazyka; neuronové sítě; FastText; SDCA; Averaged perceptron; LSTM; konvoluční sítě; předzpracování textu; ML .NET; TensorFlow

Abstract

This thesis deals with the sentiment analysis in text documents. The aim of the thesis was to describe selected methods, to implement them and to test them on appropriately selected datasets. The theoretical part deals with the issue of natural language processing and the description of methods for sentiment recognition. Levels of analysis as well as text preprocessing, possible complications and division of methods for solving this task are presented in this part. There is a also brief description of the methods, that currently achieve the best results in the sentiment classification and detailed description of selected algorithms used in the practical part of the work. All selected algorithms are in the field of machine learning and were selected according to current trends. The practical part is focused on the details of the implementation of selected methods and neural network designs, followed by the verification and comparison of all methods using various experiments. These experiments test the effect of text preprocessing, hyperparameter optimization and the effect of changes in neural network structures. To perform said experiments, three datasets were selected to validate the methods both with insufficient and excessive amount of data given.

Keywords: sentiment analysis; text classification; machine learning; natural language processing; FastText; SDCA; Averaged perceptron; LSTM; convolutional neural networks; text preprocessing; ML .NET; TensorFlow

Obsah

Seznam obrázků

Seznam tabulek

1 Úvod

Cílem této práce je vyhledat některé metody pro analýzu sentimentu v rámci textových dat, popsat tyto metody a následně je otestovat na vhodných datových sadách. Analýza sentimentu spadá do oboru zpracování přirozeného jazyka. Tato analýza se snaží zjistit, jestli daný text obsahuje nějaký názor a následně určit postoj autora textu k určité otázce. Sentiment se nejčastěji rozděluje na pozitivní a negativní.

V první kapitole se čtenář seznámí se základními pojmy a problematikou analýzy sentimentu. Dále se budeme zabývat stručným přehledem nejnovějších metod, které se v současnosti využívají pro analýzu sentimentu v textu. V následující kapitole budou podrobně popsány vybrané metody, které budou v této práci otestovány a porovnávány pomocí různých experimentů. Další kapitola se bude věnovat popisu zvolených datových sad a detailům implementace vybraných metod pro provedení experimentů. V poslední kapitole budou detailně popsány prováděné experimenty a následně budou uvedeny výsledky experimentů, jejich zhodnocení a porovnání vybraných metod.

2 Analýza sentimentu

V této kapitole bude popsána problematika analýzy sentimentu, její rozdělení podle objemu dat, příprava nestrukturovaných dat, extrakce důležitých informací, komplikace jejichž vyřešením můžeme zvýšit kvalitu klasifikace, základní rozdělení metod pro analýzu sentimentu a jejich stručný popis.

2.1 Problematika

Díky internetu máme přístup k obrovskému množství veřejných nestrukturovaných textových dat, která obsahují názory a emoce, například sociální sítě, uživatelské recenze, diskuzní fóra nebo různé blogy. Momentálně je o analýzu sentimentu velký zájem a má spoustu praktických využití. Díky ní můžeme zjistit názory veřejnosti, například o produktech, službách, obchodních značkách, politice atd. Tato data mohou být velmi užitečná pro komerční využití jako je analýza marketingu, recenze produktů, hodnocení zákaznické podpory nebo jako zpětná vazba pro produkty a služby. Jelikož jsou tato data veřejně dostupná mohou představovat výhodnější zdroj informací než průzkumy a ankety [1].

Sentiment může být v českém jazyce chápán více způsoby. V této práci bude sentiment definován jako kladná či záporná subjektivní myšlenka, názor nebo představa, jež se zakládá na pocitech, které mohou být přechodné nebo trvalé. Zjednodušeně vyjadřuje náklonnost nebo odpor.

Analýza sentimentu (angl. *Sentiment analysis* nebo *Opinion mining*) se snaží strojově identifikovat a extrahovat názory z nestrukturovaného textu. Je součástí souboru technik zpracování přirozeného jazyka (angl. *Natural language processing*, dále jen NLP). NLP se řadí do komputační lingvistiky, což je vědní obor, který se nachází na pomezí informatiky, lingvistiky a matematiky. Zabývá se zkoumáním a zpracováním přirozeného jazyka prostřednictvím počítače [2]. NLP se využívá například při strojovém překladu, získávání dat z textu nebo pro generování přirozeného jazyka.

Cílem analýzy sentimentu je najít v daném textu subjektivní sdělení, které obsahuje emoce nebo postoje a následně určit polaritu (povahu) tohoto sdělení, tedy zjistit, zda se jedná o pozitivní či negativní sdělení (popř. neutrální). Dále je potřeba nalézt cíl hodnocení, což je objekt, o kterém dané sdělení pojednává. V neposlední řadě je potřeba určit osobu nebo entitu, která daný názor vyjádřila (autora názoru), tzv. zdroj hodnocení (angl. *Opinion holder*) [3][4].

Musíme také vzít v úvahu, že analyzovaný text nemusí obsahovat žádné informace o sentimentu. To je text, který neobsahuje žádné emoce ani postoj autora, ale obsahuje pouze fakta [1]. Tento text má nulový sentiment a jeho polarita je tedy vždy neutrální. Proto jako první krok musíme určit, jestli analyzovaný text je subjektivní nebo objektivní a až poté můžeme určovat polaritu.

2.2 Úrovně analýzy

Analýzu sentimentu můžeme rozdělovat podle velikosti textu, pro který určujeme sentiment. Existují tři obecné kategorie, do kterých rozdělujeme problémy analýzy sentimentu [5].

- Analýza na úrovni dokumentu (angl. *Document-level*) – celý dokument bereme jako jednu entitu a klasifikaci sentimentu provádíme nad celým dokumentem. Jako dokument můžeme považovat i článek, recenzi nebo třeba příspěvek.
- Analýza na úrovni vět (angl. *Sentence-level*) – klasifikujeme sentiment u jednotlivých vět, a poté je uskupíme tak, abychom dostali přesnější výsledky.
- Analýza na úrovni frází (angl. *Phrase-level*) – velmi jemná analýza sentimentu, kdy analyzujeme sentiment jednotlivých slov nebo slovních spojení. Využívá se při zjišťování sentimentu jednotlivých aspektů hodnoceného předmětu. Například při hodnocení restaurace chceme vědět zvlášť hodnocení jídla, interiéru, obsluhy atd.

Před řešením problému si musíme zvolit, na jaké úrovni budeme sentiment klasifikovat. Například při analýze produktu, pozitivní hodnocení některých atributů produktu neznamená, že uživatel je s produktem kompletně spokojený. Naopak když si uživatel bude stěžovat na některé vlastnosti produktu tak to neznamená, že s produktem celkově není spokojený.

2.3 Příprava textu (Text preprocessing)

Před jakýmkoliv použitím dat, která chceme analyzovat, je potřeba data upravit a přichystat je pro následující kroky analýzy. Můžeme například rozdělit text na menší části a co nejvíce zredukovat počet celkových a unikátních slov, tím snížíme výpočetní náročnost. Pro zvýšení přesnosti klasifikace můžeme taky určit jednotlivým slovům slovní druhy a další vlastnosti.

Tokenizace je obecný popis pro metody, které rozdělují text na menší části (nejčastěji na jednotlivá slova), jež slouží jako vstup pro další algoritmy. Příklad tokenizace je tvorba n–gramů. N-gram je sekvence n po sobě jdoucích prvků (v tomto případě slov). Nejjednodušší případ jsou jednotlivá slova, tedy unigram (1-gram). Výběr optimálního čísla n záleží na jazyku a na daném problému. N-gramy nám umožňují seskupit víceslovná spojení a fráze, můžeme například spojovat podstatné jméno a přídavné jméno [6].

Stop slova (angl. *stop words*) jsou slova, které se v textu vyskytují velmi často a nenesou žádnou významovou informaci. Tato slova se liší pro každý jazyk a můžeme je z textu odstranit bez toho, abychom pozměnili výsledek klasifikace. Pro jejich odstranění potřebujeme seznam stop slov pro daný jazyk, a poté jednotlivá slova porovnáváme se slovy ze seznamu. Většinou jsou to zájmena (*já, ona, jejich, ...*), předložky (*na, při, za, ...*), spojky (*ani, ale, nebo, ...*) a v angličtině navíc členy (*a, the*). Jelikož se stop slova vyskytují v textu

velmi často, jejich odstraněním výrazně snížíme počet slov, se kterými budeme dále pracovat. Seznamy stop slov se mohou lišit podle řešeného problému. Slova, která normálně nemají žádnou přidanou hodnotu, mohou být pro některé problémy užitečné, proto není vždy vhodné tuto techniku použít [7]. Například v angličtině se často vyjadřuje zápor samostatným slovem „not“, navíc při použití lemmatizace jsou všechny zkrácené tvary záporů upraveny tak, že zápor je samostatné slovo. Toto slovo je tedy velmi frekventované, a proto se může nacházet ve slovnících stop slov.

Lemmatizace slouží pro převedení jednotlivých slov (příp. skupiny slov) do slovníkového tvaru na tzv. lemma [2]. Díky této úpravě, kdy jsou všechna slova v základním tvaru, se zredukuje celkový počet unikátních slov a zjednoduší se další práce s textem. Příklad lemmatizace může být převedení slova „odešli“ na „odejít“ nebo slova „jsme“ na tvar „být“. Podobný proces je **stematizace**, při kterém se odstraňují morfologické koncovky a předpony, abychom dostali kmen slova.

POS Tagging je proces, který přiřadí slovní druhy (angl. *Part of speech*) pro jednotlivá slova ze zpracovávaného textu. Je to důležitá část analýzy sentimentu, určení slovních druhů nám může pomoci s dalšími lingvistickými operacemi (například při lemmatizaci pro určení správného základního tvaru), ale taky je lze využít při klasifikaci sentimentu [7].

2.4 Extrakce příznaků z textů

Abychom mohli použít textová data pro algoritmy využívající strojové učení, musíme je převést do nějaké numerické reprezentace. Tento proces se nazývá Features extraction. Text dokumentu se nejprve rozdělí na jednotlivá slova, popřípadě n-gramy (viz kapitola 2.3 Tokenizace) a následně se provede transformace těchto slov do numerických vektorů stejné délky, které se nazývají příznakové vektory (angl. *features vectors*). Úprava textových dat a jejich následná vhodně zvolená numerická reprezentace, má velký vliv na výslednou kvalitu modelu. Proto existují různé techniky, jak se tato numerická reprezentace textu provádí. Níže jsou uvedeny některé z těchto technik.

2.4.1 Bag of Words (BoW)

Jeden z nejzákladnějších modelů pro numerickou reprezentaci nestrukturovaného textu používaný v NLP. Tento model reprezentuje každý textový dokument v korpusu jako příznakový vektor s pevnou délkou, kde každý příznak je unikátní slovo v korpusu. Tyto příznaky se nazývají termy. Název modelu se odvíjí od toho, že každý dokument v korpusu je zastoupen tímto vektorem bez ohledu na pořadí slov, slovní spojení a gramatiku.

Výstupem transformace je matice dokumentů a termů. Řádky matice reprezentují dokumenty a sloupce odpovídají jednotlivým termům. Prvky matice $A_{i,j}$ znázorňují ohodnocení relevance termů. Existuje více metodik, podle kterých přiřazujeme hodnoty prvkům matice. Základní

binární přístup pouze určuje, jestli se daný term v dokumentu nachází, nebo ne a podle toho jsou hodnoty vektoru čísla 1 nebo 0. Další možnost reprezentace je počítat, kolikrát se každý term v textu vyskytuje a výslednou hodnotu přiřadit danému termu. Například pokud budeme mít korpus, který obsahuje dva dokumenty (recenze):

- Recenze 1: *Obsluha byla velmi rychlá.*
- Recenze 2: *Obsluha byla v pořádku a servírka byla velmi ochotná.*

Výstupem bude matice se dvěma řádky a devíti sloupci (viz tabulka 1).

	Obsluha	**byla**	**velmi**	**rychlá**	**v**	**pořádku**	**a**	**servírka**	**ochotná**
Recenze 1	1	1	1	1	0	0	0	0	0
Recenze 2	1	2	1	0	1	1	1	1	1

Tabulka 1: Reprezentace modelu Bag of Words

Další časté rozšíření tohoto modelu je použití n-gramů. Jelikož základní BoW model nezohledňuje posloupnost slov, můžeme využít N-gram model, abychom zaznamenali slovní spojení nebo fráze. Příznakový vektor bude namísto unigramů reprezentovat například bi-gramy nebo tri-gramy. Podle studie [8] I. Kanaris a další, dosáhli velmi dobrých výsledků při filtrování spamu, použitím n-gramů s velikostí 3 a 4.

2.4.2 TF-IDF (Term Frequency-Inverse Document Frequency)

Použití modelu BoW a sestavení příznakového vektoru na základě četnosti termů, přináší několik potenciálních problémů. Větší korpusy mohou obsahovat termy, které se často opakují ve všech dokumentech a mají tendenci zastínit ostatní termy, jež mohou být důležitější pro danou úlohu. TF-IDF model je založen na předpokladu, že méně časté termy jsou důležitější a snaží se řešit tento problém pomocí normalizace a inverze četnosti. TF-IDF skóre reprezentuje důležitost termu k dokumentu v datové sadě. Termy, které se vyskytují v malém množství, jsou vyzdviženy a získají vysoké skóre. Naopak termy, které se vyskytují často jsou penalizovány. Tuto hodnotu získáme jako součin dvou metrik TF a IDF.

- TF (Term Frequency) tedy četnost daného n-gramu v dokumentu, která je následně normalizována vydělením celkovým počtem slov dokumentu.
- IDF (Inverse document frequency) je převrácená četnost n-gramů ve všech dokumentech. Udává nám důležitost slova v datové sadě, čím častěji se slovo vyskytuje v datové sadě, tím méně je důležité. IDF se vypočítá podle vzorce (1), kde N je celkový počet dokumentů a df_t je počet dokumentů, ve kterém se daný term vyskytuje.

$$idf_t = log\frac{N}{df_t} \tag{1}$$

2.4.3 Vnoření slov (Word Embeddings)

BoW model nezachycuje žádné informace o kontextu a významu slov, a navíc produkuje velké řídké vektory. Vnoření slov pomocí prediktivních metod je efektivní reprezentací textu, kde slova, která mají stejný význam, mají podobnou reprezentaci. Ve skutečnosti se jedná o skupinu technik, kde jsou jednotlivá slova reprezentována jako vektory s reálnou hodnotou, v předdefinovaném vektorovém prostoru. Každé slovo je mapováno na jeden vektor a hodnoty vektoru jsou určeny způsobem, který se podobá neuronovým sítím. Všechny slova mají obvykle na počátku přiřazen náhodný vektor a poté jsou postupně upravovány tak, aby dosáhly požadovaných vlastností. Tyto vektory uchovávají velkou část sémantických i morfologických vlastností slov. Pro ukázku použijeme větu *„Obsluha byla velmi rychlá“*. Všechny slova budou reprezentovány jako 4dimenzionální vektory (viz tabulka 2).

obsluha	1,2	-0,1	4,3	3,2
byla	0,5	2,4	-0,8	0,6
velmi	1,9	0,5	-0,2	-0,8
rychlá	0,9	1,3	2,8	1,4

Tabulka 2: Ukázka vektorové reprezentace word embeddings s velikostí vektoru 4

Jelikož se jedná o numerické znázornění kontextových podobností mezi slovy (což mohou být rody, geografie, slovesný čas, nebo něco úplně jiného) ve vektorovém prostoru, můžeme s těmito vektory provádět jednoduché aritmetické operace. Označíme-li si vektorovou reprezentaci slova jako *vektor(slovo)*, můžeme provádět operace typu *vektor(otec) - vektor (muž) + vektor(žena)*, výsledný vektor bude velmi podobný jako *vektor(matka)*. Tyto vztahy jsou jazykově nezávislé. Přímá vzdálenost slov není úplně vhodná pro porovnání podobnosti, neboť může být značně ovlivněna četností slova v korpusu. Vhodnou metrikou vzdálenosti vektorů je například kosinová podobnost.

Pro vytvoření kvalitního modelu vnoření slov je potřeba velké množství textu (miliony až miliardy slov). Proto výzkumníci často používají předtrénované volně dostupné modely. Tyto modely bývají trénované na obrovských datových sadách a nejsou doménově závislé. Lze je využít tak jak jsou bez dalších úprav nebo je můžeme přeučit s menší datovou sadou, aby lépe vyhovovaly řešenému problému. Nejznámější předtrénované modely jsou word2vec, GloVe a FastText.

2.5 Komplikace

Analýza sentimentu je složitý problém, který se liší v závislosti na doméně a zdroji dat. Při snaze o zvýšení přesnosti klasifikace sentimentu se můžeme setkat s různými komplikacemi, které budeme muset vyřešit. Například při klasifikaci příspěvků na sociální síti se setkáme s hovorovými výrazy, překlepy nebo emotikony, které lze využít pro přesnější klasifikaci.

Další problém, na který můžeme narazit jsou srovnávací výrazy. Takový výraz srovnává hodnocený objekt s jinými objekty podobného typu. Například věta „*Coca-cola chutná líp než Pepsi.*“ je srovnávací výraz. Tyto výrazy můžeme detekovat a určovat, který produkt autor upřednostňuje [9].

Pro správnou klasifikaci sentimentu je důležité správně vyhodnotit negaci v klasifikovaném textu, jelikož může úplně změnit výslednou polaritu, ale není to pravidlo. Negace ve větě „*Tento nápoj není dobrý.*“ jasně otočí polaritu z pozitivní na negativní. Zatímco věta „*Není to nejlepší nápoj.*“ neznamená že je nápoj špatný, tedy věta nemusí být negativní. Další problém, který může nastat je dvojitá negace, negace se mohou navzájem vyrušit, ale není to tak vždy. Například ve větě „*Není film, který by to nezvládlo přehrát.*“, jedna negace vyruší druhou a polarita zůstane kladná. Naopak ve větě „*Nikdo nehodnotí výrobek dobře.*“ dvojitá negace zápor nevyruší, ale spíše ho stupňuje.

Jeden z velmi složitých problémů při analýze sentimentu je sarkasmus a v některých případech může velmi ovlivnit kvalitu analýzy sentimentu. Sarkasmus mění polaritu z pozitivní na negativní a naopak. Strojová detekce sarkasmu je další z úloh v oboru NLP. U každého výrazu záleží, v jakém je kontextu. Jeden výraz může mít při různém kontextu různou polaritu [10].

Vyřešením těchto problémů můžeme zvýšit přesnost klasifikace sentimentu, ale vždy je třeba mít na paměti, že všechny algoritmy pro analyzování sentimentu pracují s určitou chybou. Jedním z problémů je, že ani člověk nedokáže vždy správně určit sentiment daného textu (například kvůli sarkasmu a ironii) a názory více lidí se mohou lišit.

2.6 Algoritmy

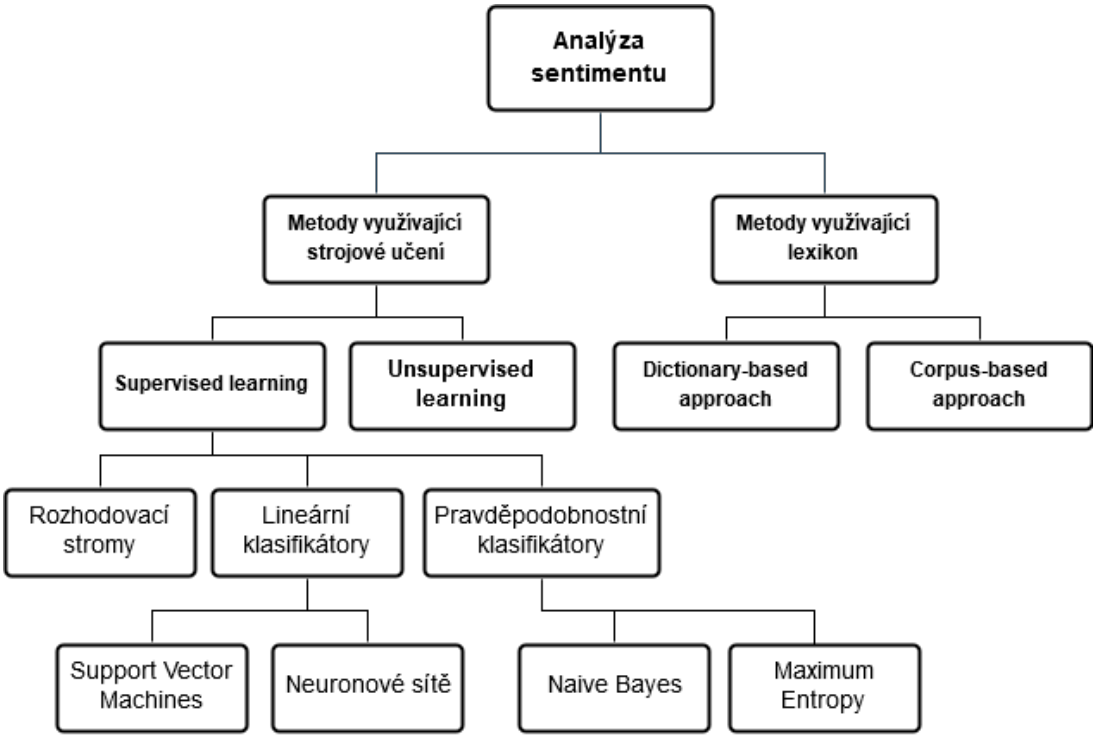
Existují různé metody a algoritmy pro analýzu sentimentu, jejich základní rozdělení je následující:

- Metody využívající lexikon (angl. *lexicon based approaches)*
- Metody využívající strojové učení (angl. *machine learning approaches)*
- Kombinovaný přístup (angl. *hybrid approach)*

Metody využívající lexikon závisí na slovnících, podle kterých se určuje sentiment jednotlivých vět či výrazů. Slovník je kolekce slov a výrazů vyjadřujících sentiment a každému slovu nebo výrazu je přiřazena síla sentimentu (pozitivní nebo negativní). Klasifikace sentimentu u těchto metod je založena na principu sečtení síly sentimentu jednotlivých slov nebo frází analyzovaného textu a podle výsledné síly se určí sentiment. Přesnost klasifikace u tohoto přístupu je ovlivněna hlavně použitým slovníkem. Již vytvořené slovníky jsou velmi doménově závislé, proto je třeba vytvořit nový slovník nebo upravit již stávající. Vytvořit slovník manuálně by bylo velmi zdlouhavé a náročné, proto se používají automatické metody (Dictionary-based approach) [11].

Metody využívající strojové učení se dále rozdělují na supervised a unsupervised metody. Supervised metody (učení s učitelem) používají data, která mají anotace určující třídu sentimentu. Tato data se rozdělí na trénovací data a testovací data (obvykle v poměru 8:2). Z trénovacích dat se vytvoří model podle identifikovaných příznaků a vlastností (angl. *features)*, jehož přesnost a další metriky se určují na testovacích datech. Nejznámější algoritmy pro klasifikaci jsou rozhodovací stromy, SVM (Support vector machines), neuronové sítě, Naïve Bayes a Maximum Entropy [11]. Unsupervised metody (učení bez učitele) se v oboru NLP používají tam, kde je obtížné získat trénovací data obsahující anotace. Pro analýzu sentimentu se příliš nepoužívají.

Kombinovaný přístup se snaží zkombinovat slovníkové přístupy a strojové učení pro dosažení co nejvyšší přesnosti klasifikace. Tento přístup není příliš běžný.



Obrázek 1: Rozdělení metod klasifikace [11]

3 Současné metody (State of the art)

Analýze sentimentu se v posledních letech dostává velké pozornosti. Textová data jsou většinou nestrukturovaná, obsahují mnoho šumu a vytáhnout z nich potřebné informace může být složité a výpočetně náročné. Proto je potřeba stále vyvíjet efektivnější metody pro analýzu sentimentu. Aktuálně je velmi populární hluboké učení (angl. *deep learning)* a dosahuje vysoké úspěšnosti. Využívá neuronové sítě s velkým počtem vrstev, hodí se spíše tam, kde máme značné množství trénovacích dat (až miliony prvků) a jsou výpočetně velmi náročné. Dále budou stručně popsány nejnovější techniky, pomocí kterých lze dosáhnout vysokou přesnost klasifikace sentimentu.

3.1 Universal Language Model Fine-tuning for Text (ULM-FIT)

ULM-FIT je metoda umožňující přenos naučeného modelu pro jakýkoliv úkol z oboru NLP a dosažení skvělých výsledků bez nutnosti trénovat model od nuly. Tato metoda zahrnuje doladění modelu (angl. *fine-tuning*) a před-trénovaný jazykový model. S použitím pouze 100 anotovaných (labelled) vzorků, je možné dosáhnout stejné kvality klasifikace jako se 100krát více daty při trénování modelu od nuly [12]. ULM-FIT používá jednotky LSTM (Long short-term memory), neboli rekurentní neuronovou síť. Celý proces se skládá ze tří částí:

- **Před-trénování doménově nezávislého modelu** – Model je před-trénován na datech Wikitext_103_dataset[1], skládající se z 28,595 strukturovaných anglických článků z Wikipedie. Tento krok je výpočetně velmi náročný.
- **Doladění jazykového modelu podle daného úkolu** – Model se doladí pomocí dat z oblasti řešeného úkolu a zdokonalí klasifikační model.
- **Doladění klasifikátoru** – Poslední část zahrnuje trénování modelu s přidáním dalších dvou lineárních bloků.

3.2 BERT

BERT (Bidirectional Encoder Representations for Transformers) je open source technika hlubokého učení pro zpracovaní přirozeného jazyka, používající obousměrné trénování transformátoru a populární attention model, pro modelování jazyka [13]. Aktuálně je velmi populární a je to jedna z nejlepších metod pro různé problémy NLP včetně analýzy sentimentu. Model je před-trénovaný na 330 milionech slov z anglické wikipedie a BooksCorpus[2]. Existuje více modelů, ale všechny se odvíjejí od dvou základních.

- **BERT base** – základní model má 12 kódovacích vrstev (angl. *transformer Blocks*), 768 skrytých vrstev, 12 attention heads se 110 miliony parametrů.

[1] https://blog.einstein.ai/the-wikitext-long-term-dependency-language-modeling-dataset/
[2] https://arxiv.org/abs/1506.06724

- **BERT large** – větší model, který dosahuje nejlepších výsledků, se skládá z 24 kódovacích vrstev, 1024 skrytých vrstev a 16 attention heads s celkovým počtem 340 milionů parametrů.

BERT je unsupervised technika (učení bez učitele), což znamená, že se učí pouze pomocí prostého textu korpusu, a tedy nepotřebuje anotovaná data. Vychází z metod, jako jsou ELMo, ULMFit nebo Semi-supervised Sequence Learning s tím rozdílem, že při trénování využívá metodu Masked Language Model (MLM), která náhodně maskuje 15 procent vstupních tokenů před tím, než prochází obousměrným kódovacím transformátorem a díky tomu je trénovaný model jako jediný hluboce obousměrný (angl. *deeply bidirectional*). Pro doladění modelu už stačí přidat pouze jednu výstupní vrstvu, tento proces je výpočetně levný a můžeme tak jednoduše upravit model pro řešený problém.

3.3 XLNet

XLNet patří v současnosti mezi další nejúspěšnější metody pro různé úkoly z oblasti NLP. Řádí se mezi před-trénované neuronové sítě a funguje na stejném principu jako předchozí metoda. Zatímco metoda BERT zanedbává závislost mezi maskovanými pozicemi a trpí rozporem mezi před-trénováním a dolaďováním modelu, XLNet má všechny výhody metody BERT bez jejích nevýhod. Využívá principy autokódovacích (angl. *autoencode*) metod, určených pro trénování obousměrného modelu bez anotovaných dat (unsupervised learning), přičemž se zbavuje jejich limitací pomocí auto regresivní formulace. Základ této metody je převzat z metody Transformer-XL[3], přesněji mechanismus rekurentního segmentu a relativní kódovací schéma [14].

[3] https://arxiv.org/abs/1901.02860

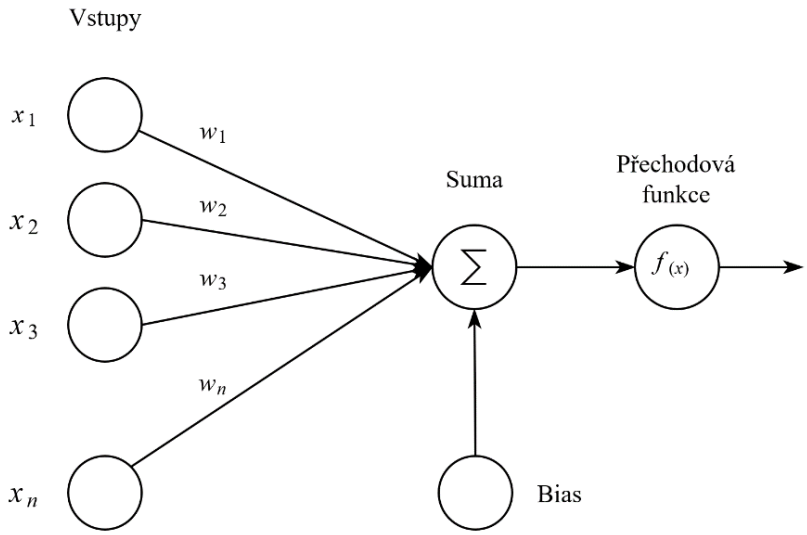
4 Klasifikační metody

V poslední době jsou pro analýzu sentimentu velmi populární metody strojového učení a dosahují vysoké přesnosti klasifikace. Tato práce se zabývá pouze metodami strojového učení, které jsou v následujících podkapitolách detailněji popsány.

4.1 Averaged perceptron

Perceptron je metoda typu učení s učitelem pro binární klasifikaci, která mapuje vektor příznaků (angl. *features vector*) $x = \{x_0, x_1, ..., x_n\}$ na výstupní hodnoty $f(x)$. Pro predikci hledá oddělující nadrovinu (angl. *hyperplane*) mezi dvěma množinami. Perceptron se skládá ze tří částí (viz obr. 2):

1. Vstupní hodnoty $\{x_0, x_1, ..., x_n\}$
2. Váhy $\{w_0, w_1, ..., w_n\}$ a Bias
3. Přechodová funkce $f_{(x)}$



Obrázek 2: Schéma perceptronu

Přechodová funkce (aktivační funkce) slouží k mapování vstupních hodnot na výstupní hodnotu dle vybrané přechodové funkce (například hodnoty 0 až 1). Hodnoty vah určují směrnici dělící nadroviny a Bias ji umožňuje posouvat. Učení perceptronu má následující fáze:

1. Váhy jsou na začátku algoritmu nastaveny náhodně nebo na hodně nízké hodnoty.
2. Každý vstup se vynásobí jeho váhou, výsledné hodnoty se sečtou, a ještě přičteme Bias.

3. Na výslednou hodnotu aplikujeme aktivační funkci.

4. Výstupní hodnota se porovná se správnou hodnotou, pokud se liší je potřeba upravit váhy.

5. Váhy se upraví podle následujícího pravidla: $w_{t+1} = w_t + e \cdot p$, přičemž w_t je aktuální váha, e je chyba, kterou spočítáme jako *zjištěný výsledek – správný výsledek* a p je učící konstanta (hyperparametr). Bias se upravuje následovně: $b_{t+1} = b_t + e$.

Kroky 2-5 se opakují pro všechna trénovací data. Averaged perceptron algoritmus během učení všechny váhy a bias průměruje a na konci učení se výslednému modelu přiřadí tyto průměrné váhy. Tímto se zabrání problému klasického perceptronu, kdy chyby v pozdější fázi učení měly velký dopad na váhy a model byl náchylný k přeučení.

4.2 Stochastic Dual Coordinate Ascent (SDCA)

Metoda SDCA kombinuje nejlepší vlastnosti a schopnosti logistické regrese a SVM (Support Vector Machine). Má lineární míru konvergence (angl. *convergence rate*) a vysoký empirický výkon pro binární klasifikační problémy (např. analýzu sentimentu, kde rozhodujeme, jestli je sentiment pozitivní nebo negativní) [15].

Je to supervised metoda strojového učení, spadající do kategorie lineárních klasifikátorů. Používá se pro problémy formulované jako minimalizace konvexních loss funkcí (angl. *convex loss functions*), tedy minimalizace loss funkcí, jejichž lokální minimum je zároveň globální minimum. Loss funkce v oboru strojového učení je metoda používaná při trénování modelu, konkrétněji je to způsob ohodnocení výstupu pro následnou úpravu modelu tak, aby se snížila chybovost klasifikace.

SDCA provádí iterativní aktualizace náhodně vybraných souřadnic s cílem maximalizovat *dual objective*, tzn. hledáme nejlepší společné řešení pro dvě funkce. Má dvě různé proměnné, které se vztahují ke každému prvku v trénovacích datech, v každé iteraci se optimalizuje *dual objective* s ohledem na jednu z těchto proměnných. Po každé iteraci zjišťujeme tzv. *duality gap*, když je tato hodnota dostatečně malá, tak se algoritmus ukončí [15].

Pro tuto práci byla použita implementace SDCA metody s několika úpravami. Základní SDCA algoritmus používá sekvenční náhodný přístup k trénovacím datům v paměti, což omezuje použití pro velké objemy dat (vysoká paměťová a výpočetní náročnost). Použitá implementace využívá paralelizaci a data načítá z disku po blocích, tudíž není omezená velikostí datové sady [16]. Je založena na logistické regresi a jako loss funkce je použita Logistic Loss, která je zde definována následovně:

$$f(x, y) = log(1 + e^{-yx}) \tag{2}$$

Výstupem je pravděpodobnost (0 až 1), určující, do které třídy klasifikovaný prvek spadá, poté se zvolí třída (pozitivní, negativní) podle toho, ke které hodnotě se pravděpodobnost více blíží.

4.3 FastText

FastText model je rozšíření staršího modelu Word2Vec[4]. Prediktivní modely obvykle považují každé slovo za samostatnou entitu a generují pro ně tzv. word embeddings. To však představuje vážné omezení u jazyků s obrovskými slovníky a mnoha vzácnými slovy, která se vyskytují pouze ojediněle. Model FastText považuje každé slovo jako skupinu podslov, neboli n-gramů tvořených z písmen daného slova. Hlavní výhoda této metody spočívá v rychlosti trénování modelu, které je mnohonásobně rychlejší než u hlubokých neuronových sítí, zatímco přesnost vytrénovaného modelu je srovnatelná [17].

Vektory reprezentující slova se tvoří tak, že se na začátek a konec slova přidají hraniční symboly „<“ a „>“. To pomáhá rozlišovat význam kratších slov a n-gramů znaků, vzniklých při tvorbě podslov. Pokud si vezmeme slovo „výroba“ a zvolíme $n = 3$ tak slovo bude reprezentováno n-gramy: *<vý, výr,ýro, rob, oba, ba>* a přidáme celé slovo *<výroba>*. Díky hraničním symbolům je možné rozlišit n-gram *výr* a samostatné slovo *<výr>*. Výsledný vektor pro slovo „výroba“ bude suma vektorů všech těchto n-gramů. Cílem rozdělení slov na kratší podslova je zlepšit reprezentaci vzácných slov. I když se tyto slova vyskytují málo, tak jejich podslova se mohla vyskytnout v jiných slovech. Navíc z jednotlivých podslov mohou vzniknout vektory pro slova, která se ani nenacházejí v trénovacím korpusu.

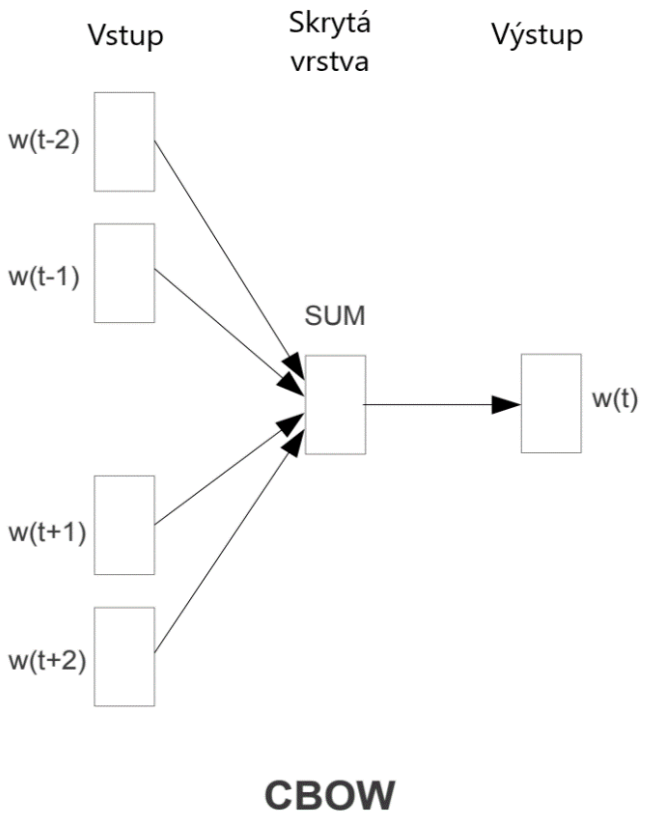
FastText podporuje dvě různé architektury modelu pro vytvoření vektorové reprezentace, a to model Continuous Bag og Words (CBOW) a Skip-gram model. Obě architektury jsou založeny na neuronových sítích a pocházejí z modelu Word2Vec. K učení používají lokální kontext každého slova, který je definován oknem sousedících slov. Velikost tohoto okna je konfigurovatelný parametr obou modelů.

CBOW model se snaží predikovat aktuální slovo podle jeho kontextu (okolních slov). Tato mělká neuronová síť se skládá ze 3 vrstev (viz obr. 3). Vstupní vrstva se skládá z n vektorů o velikosti v, kde n je velikost okna kontextu a v je velikost slovníku. Počet neuronů v prostřední skryté vrstvě určuje velikost vektorů, reprezentujících jednotlivá slova. Výstupní vrstva má pouze jeden výstupní vektor o velikosti v, je to tzv. One hot reprezentace slov.

Skip gram model se naopak snaží určit obklopující slova na základě jednoho slova. Architektura modelu je stejná jako v případě modelu CBOW s tím rozdílem, že je přehozená vstupní a výstupní vrstva (viz obr. 4). Na vstupu tedy máme jedno slovo a na výstupu dostaneme obklopující slova. Na konci výstupní vrstvy je použita aktivační funkce Softmax, takže každý prvek výstupního vektoru popisuje, jak je pravděpodobné, že se konkrétní slovo objeví v kontextu daného slova. Skip-gram model funguje lépe při menší velikosti trénovacích dat a lépe reprezentuje vzácná slova. CBoW model se trénuje rychleji než Skip-gram model a lépe reprezentuje častá slova.

Pro klasifikaci FastText používá model CBOW s tím rozdílem, že místo predikce slova podle kontextu, predikuje kategorii (třídu) podle kontextu. Jako cost funkci v trénovaném modelu

[4]https://arxiv.org/abs/1301.3781



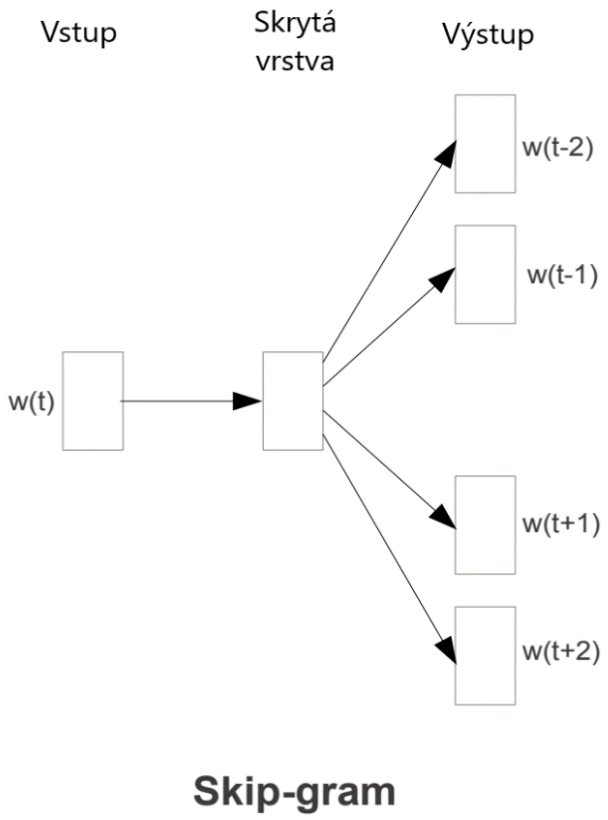
Obrázek 3: Architektura modelu CBOW [18]

můžeme použít funkci Softmax, Hierarchický softmax nebo Negativní vzorkování.

- **Softmax** – Tato aktivační funkce se často používá při klasifikaci do více tříd, v poslední vrstvě klasifikátorů postavených na neuronových sítích. Mění numerické výstupy poslední vrstvy na pravděpodobnosti, suma těchto pravděpodobnostní je rovna 1. Výpočetní složitost je $O(k \cdot h)$, kde k je počet predikovaných tříd a h je velikost slovníku. Jednotlivé pravděpodobnosti se vypočítají podle rovnice (3).

$$s(x_i) = \frac{e^{x_i}}{\sum_{j=1}^{n} e^{x_j}} \tag{3}$$

- **Hierarchický Softmax** – Aproximační funkce pro plný Softmax umožňuje výpočetně efektivnější trénování modelu pro velké množství tříd. Cílem je vytvořit binární strom, jehož listy odpovídají třídám. Vnitřní uzly udávají relativní pravděpodobnost svých potomků, která se mění při trénování modelu. Pravděpodobnost výstupu funkce je produkt pravděpodobností všech uzlů od kořene stromu až po list. FastText používá při tvorbě stromu Huffmanovo kódování, tedy třídy, které mají větší pravděpodobnost, jsou blíže u kořene stromu. Díky tomu je průchod stromem pro nejčastější výstupy daleko rych-



Obrázek 4: Architektura modelu Skip-gram [18]

lejší [18]. Použitím této funkce můžeme dosáhnout výpočetní složitosti až $O(h \cdot log(k))$. Cena za nižší výpočetní složitost je nepatrné snížení přesnosti modelu. Tato funkce se vyplatí použít, když máme nevybalancovanou datovou sadu nebo pokud mámě hodně klasifikačních tříd [18].

- **Negativní vzorkování** – Pokud budeme predikovat velké množství tříd, ale budeme mít stejný počet trénovacích dat pro každou třídu, tak je vhodné použít funkci Negativní vzorkování [18]. Místo úpravy všech vah pro každý vstup, náhodně vybereme malý počet „negativních vzorků“ (většinou 2-20, podle velikosti datové sady) a upravíme váhy jen pro vstup a negativní vzorky. Negativní vzorky (při klasifikaci) jsou třídy, pro které chceme, aby byl výstup negativní.

4.4 Umělá neuronová síť

Díky stále se zvyšující popularitě neuronových sítí a hlubokého učení v posledních několika letech, byla oblast zpracování přirozeného jazyka (dále jen NLP) posunuta hodně vpřed. Více než deset let se používaly především základní lineární modely strojového učení, jako Perceptron,

SVM nebo Logistická regrese, jež pro trénovaní používaly velmi řídké mnoho dimenzionální vektory. Okolo roku 2014 se v oboru NLP začalo přecházet z těchto lineárních modelů na nelineární neuronové sítě, které naopak pro vstupy používaly husté vektory [19]. Zatímco některé sítě jsou pouze jednoduchým zobecněním lineárních modelů, jiné jsou velmi pokročilé, fungují na různých principech a poskytují nové možnosti trénování modelu. Mimo NLP se neuronové sítě používají například v oblasti zpracování obrazu, komprese a k predikci časových řad. Obecně je lze použít pro klasifikaci, predikci, shlukování, vizualizaci atd. V této práci se dále budeme zabývat pouze klasifikací.

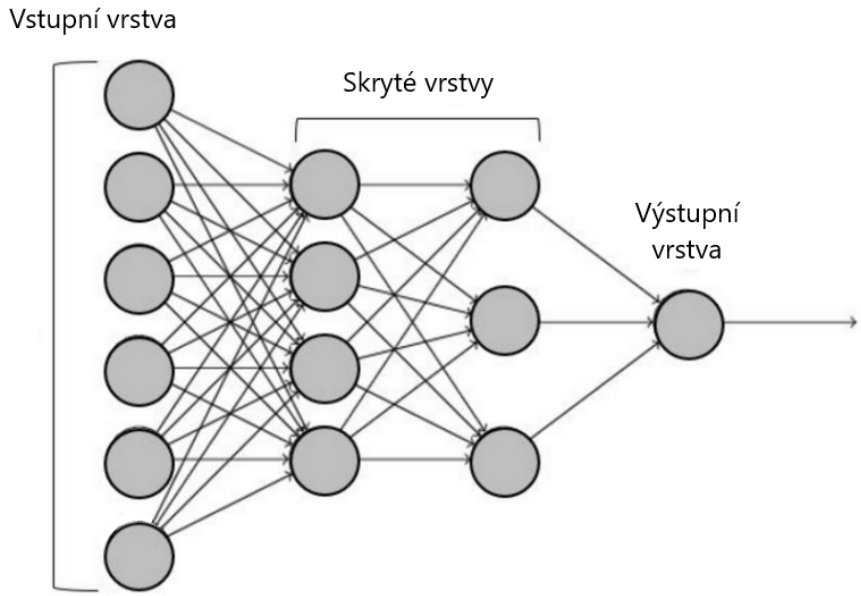
Neuronová síť je řadou algoritmů, které se snaží rozpoznat základní vztahy a vzory mezi daty, prostřednictvím procesů napodobující fungování lidského mozku. Jeden umělý neuron (perceptron) může rozdělit množinu možných řešení pouze lineárně, proto je potřeba pro složitější úlohy sestavit síť neuronů, kde každý neuron v síti řeší pouze část celého problému. Neuronové sítě se skládají ze vzájemně propojených neuronů, z nichž každý má určitý počet vstupů a pouze jeden výstup. Tyto neurony jsou organizovány ve více vrstvách, každá síť obsahuje jednu vstupní a výstupní vrstvu. Mezi nimi se může nacházet jedna nebo více skrytých vrstev, pokud síť obsahuje hodně skrytých vrstev, označuje se jako hluboká neuronová síť. Jednotlivé vrstvy mohou obsahovat libovolný počet neuronů, které sdílejí některé vlastnosti (např. aktivační funkci). Neurony si mezi sebou předávají signály a transformují je pomocí přenosových (aktivačních) funkcí. Vstupem neuronu mohou být vstupní data nebo výstup z jiného neuronu. Tato spojení jsou vážené hrany, jejichž hodnoty se mění v průběhu učení. Vstupní data musí být numerické hodnoty ve vektorovém prostoru, ať už jde o obrázky, zvuk nebo text je potřeba tyto data transformovat do číselné reprezentace.

Jedna z nejpoužívanějších neuronových sítí je dopředná síť (angl. *feed-forward network*). Tato síť je tvořena několika vrstvami neuronů, přičemž každý neuron jedné je vrstvy vzájemně propojen s každým neuronem následující vrstvy (viz obr. 5). Skládá se ze tří a více vrstev, řadí se do kategorie učení s učitelem. Účelem dopředných sítí je aproximace funkce *f*. Například pokud budeme mít klasifikátor $y = f_{(x)}$ mapující vstup x do kategorie y. Dopředná síť definuje mapování $y = f_{(x,\theta)}$ kde θ je hodnota, která se upravuje učením pro dosáhnutí co nejlepší aproximace [20].

Jak název napovídá, vstupní data prochází sítí pouze dopředu, od vstupní vrstvy k výstupní. Počet vstupních neuronů se odvíjí od počtu vstupů matematického modelu. Každý vstup má přidruženou váhu. Funkce neuronu se skládá z následujících kroků:

- Neuron sečte všechny součiny vstupních hodnot a jejich vah.
- Na výslednou hodnotu se aplikuje aktivační funkce.
- Výstup aktivační funkce může být vstup dalších neuronů, nebo výstup sítě.

V průběhu učení se snižuje hodnota cost funkce, až do stavu, kdy se asymptoticky blíží k nule. Pokud je výsledek chybové funkce roven nule, tak se ze sítě stal lineární klasifikátor, síť je poté



Obrázek 5: Struktura dopředné sítě

přeučená (angl. *overfitted*) a ztrácí abstrakci, tedy není schopná správně klasifikovat jiná data než trénovací.

4.4.1 Učení neuronových sítí

Učení je proces optimalizace vah, při kterém se minimalizuje chybovost predikce a síť dosahuje lepší přesnosti klasifikace. Zjednodušeně řečeno, všechny trénovací metody opakovaně počítají odhad chyby pro trénovací data (rozdíl výstupu sítě a očekávané hodnoty), vypočítávají gradient s ohledem na chybu a upravují parametry sítě ve směru gradientu. Metody se liší v tom, jak počítají odhad chyby a jak upravují parametry ve směru gradientu. K odhadu chyby se používají cost funkce (někdy nazývané loss funkce) a pro úpravu parametrů tzv. optimalizační algoritmy. Gradient nám říká, jak se mění chyba klasifikace, když se mění váhy sítě. Abychom efektivně vypočítali gradient pro celou síť, běžně se používá technika zvaná zpětná propagace (angl. *backpropagation*).

Každý neuron v síti generuje nějakou chybu. Tato chyba ovlivňuje ostatní neurony a společně ovlivňují celkovou chybu sítě. Chybu každého neuronu můžeme spočítat parciální derivací cost funkce podle váženého vstupu daného neuronu (4).

$$\delta = \frac{\partial C}{\partial n} \tag{4}$$

Zpětná propagace prochází síť od poslední vrstvy na začátek a počítá chybu všech neuronů ve výstupní a skryté vrstvě. Výpočet chyby se liší podle toho, v jaké vrstvě se neuron nachází.

Pro výpočet chyby neuronu ve výstupní vrstvě, pomocí řetězového pravidla, použijeme následují rovnici:

$$\frac{\partial C}{\partial w_{ij}} = \frac{\partial C}{\partial o_j}\frac{\partial o_j}{\partial n_j}\frac{\partial n_j}{\partial w_{ij}} \tag{5}$$

kde o_j je výstup z neuronu j, n_j je suma vážených vstupů neuronu j před použitím aktivační funkce a w_{ij} je váha vstupu i, neuronu j. Výpočet chyby neuronu ve skryté vrstvě je o něco složitější. U neuronu ve výstupní vrstvě známe výstup, ale pro neuron ve skryté vrstvě ho musíme dopočítat. V první části předchozí rovnice musíme brát v úvahu, že cost funkce počítá se všemi neurony, jejichž vstupy jsou výstupy neuronu j. Tuto množinu neuronů si označíme jako L, potom se první část rovnice vypočte následovně:

$$\frac{\partial C}{\partial o_j} = \sum_{i \in L}\left(\frac{\partial C}{\partial o_l}\frac{\partial o_l}{\partial n_l}w_{jl}\right) \tag{6}$$

Když spočteme chybu všech neuronů, můžeme upravit jednotlivé váhy. Výslednou váhu dostaneme použitím rovnice (7), kde η je parametr *learning rate.* Tento parametr redukuje velikost kroku techniky gradient descent. Pokud je příliš vysoký, tak budou změny příliš velké a nemusíme nikdy najít minimum funkce. V opačném případě bude učící algoritmus velmi zpomalen. Celá rovnice je vynásobena -1, což nám zaručí, že nová váha sníží chybu sítě.

$$\Delta w^{ij} = -\eta\frac{\partial C}{\partial w_{ij}} \tag{7}$$

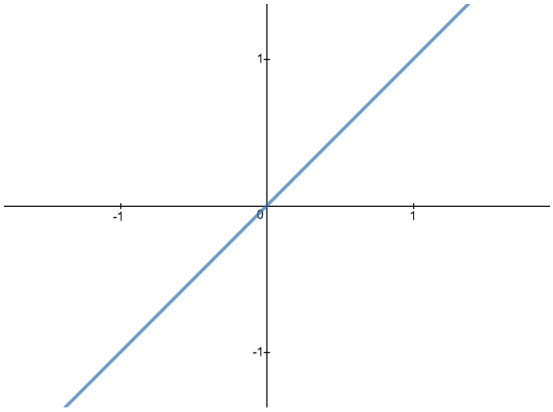
4.4.2 Aktivační funkce

Aktivační funkce se používá na výstupu neuronů. Jejich účelem je konvertovat vstupní signál neuronu na výstupní signál tak, abychom přidali síti nelineární vlastnosti. Obecně platí, že čím máme složitější data ze kterých chceme síť učit, tím více nelineární mapování potřebujeme. Bez aktivační funkce není možné provést složité mapování matematicky. Výběr vhodné aktivační funkce neuronu má vliv na konvergenci neuronové sítě. Níže jsou nejčastěji používané funkce popsány.

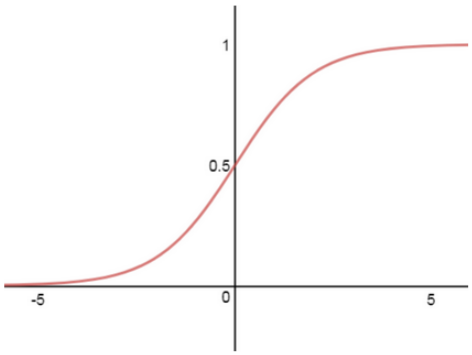
- **Lineární funkce** – Pokud neaplikujeme žádnou transformaci, tak dostaneme lineární funkci (8).

$$f_{(x)} = x \tag{8}$$

  Neuron by provedl pouze sumaci vážených vstupů, přidal bias a tento výsledek by byl výstup neuronu. Taková neuronová síť je v podstatě jen lineární regresní model. Pro složitější problémy je potřeba, aby síť byla schopna aproximovat nelineární vztahy mezi vstupem a požadovaným výstupem.



Obrázek 6: Graf lineární funkce



Obrázek 7: Graf logistické funkce

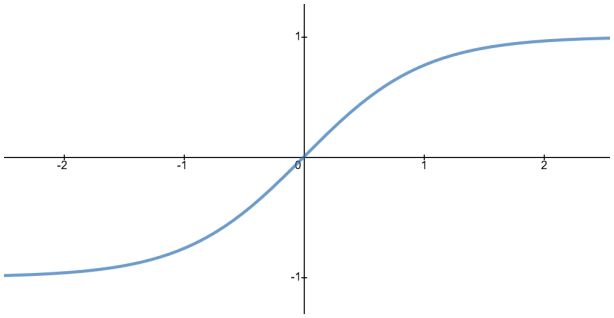
- **Logistická funkce** (sigmoida) – Aktivační funkce, která transformuje reálné číslo x na hodnotu v intervalu $(0, 1)$. Je jednoduchá na použití, ale má spoustu nevýhod, kvůli kterým není příliš populární. Výstup funkce není centrován okolo nuly (viz obr. 7), to způsobuje, že úpravy gradientu jsou příliš velké a v různých směrech. Taky konvergence této funkce je příliš pomalá. Její definice je následující.

$$f_{(x)} = \frac{1}{1 + e^{-x}} \tag{9}$$

- **Tanh** (hyperbolický tangens) – Výstup funkce nabývá hodnot v intervalu *(-1, 1)*, tudíž je centrovaný kolem nuly. Tímto řeší problém sigmoidy a optimalizace této funkce je mnohem jednodušší. Definicí funkce je rovnice (10).

$$f_{(x)} = \frac{1 - e^{-2x}}{1 + e^{-2x}} \tag{10}$$

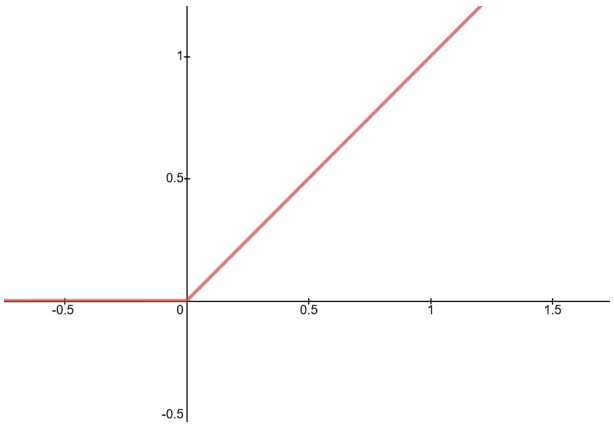
- **ReLu** (Rectified linear unit) – Velmi populární funkce, jejíž definice je (11). Transformuje všechny negativní hodnoty na nulu (viz obr. 9). Její hlavní výhody jsou nízká výpočetní



Obrázek 8: Graf Tanh funkce

náročnost a jednoduchá derivace, taktéž urychluje konvergenci směrem ke globálnímu minimu loss funkce. Předchozí dvě funkce se při vysoké hodnotě blíží k 1, jejich gradient je poté téměř nulový, a to vede k tomu, že se celkový gradient začne blížit k nule a učící proces se téměř zastaví. ReLU tento problém nemá, čímž je tato funkce vhodná speciálně pro vícevrstvé sítě, které jsou náchylné k problémům s „mizejícím“ gradientem. Používá se pouze ve skrytých vrstvách. Konverze negativních hodnot na nulu může být taky nevýhoda této funkce. Při převaze negativních hodnot se začne gradient velmi blížit k nule a přestanou se upravovat váhy neuronů. Neurony, které se dostanou do takového stavu, přestanou reagovat na odlišné vstupy a nebudou se dále učit. Těmto neuronům se říká „mrtvé neurony“, protože jejich výstup je vždy nula. Je běžné že 20–50 % všech neuronů v síti, které používají aktivační funkci ReLU jsou mrtvé neurony. Existují různé variace ReLU, které se snaží tento problém zmírnit (např. Leaky ReLU).

$$f_{(x)} = \begin{cases} 0, & x < 0 \\ x, & x \geq 0 \end{cases} \tag{11}$$



Obrázek 9: Graf funkce ReLu

4.4.3 Optimalizační algoritmy

Optimalizační algoritmy se v neuronových sítích používají k minimalizaci cost funkce. Po každém trénovacím cyklu provádí úpravy vah a biasu, dokud cost funkce nedosáhne globálního minima. Existuje mnoho takových algoritmů, většina z nich vychází ze základního optimalizátoru gradient descent. Níže jsou popsány některé z nich.

- **Stochastic gradient descent** (SGD) - SGD je varianta klasické metody gradient descent (GD). GD upravuje parametry modelu pouze jednou za celý průchod trénovacími daty, což může být velmi pomalé a s velkou datovou sadou mohou nastat problémy s pamětí. SGD se snaží řešit tento problém častější úpravou parametrů. Parametry modelu jsou upraveny pro každý prvek trénovacích dat a není potřeba ukládat výsledky cost funkce, proto je SGD mnohem rychlejší a má menší paměťovou náročnost. Kvůli častým úpravám mají parametry velké odchylky a způsobují kolísání cost funkce v různých intenzitách. To pomáhá nalézt nová lokální minima, ale v konečném důsledku také komplikuje konvergenci v minimu a dokonce může algoritmus změnit hodnotu po najití globálního minima.

- **Adam** (Adaptive Moment Estimation) – Metoda Adam využívá Momentum (k určení gradientu využívá gradienty z předchozích kroků) a adaptivní learning rate pro rychlejší konvergenci. Udržuje rozdílné parametry *learning rate* pro každou váhu sítě a každý parametr je samostatně upravován podle vývoje učení. Ke změně parametrů learning rate počítá exponenciální průměr gradientů a druhé mocniny vah gradientů pro každý parametr. Definice rovnic pro tzv. první moment m_t (průměr) a druhý moment vt (necentrická odchylka) je následující:

$$m_t = \beta_1 \, v_{t-1} + t(1-\beta_1)g_t \tag{12}$$

$$vt = \beta_2 \, v_{t-1} + (1-\beta_2)g_t^2 \tag{13}$$

  Hodnoty m_t a vt jsou inicializovány jako nulové vektory, β_1 a β_2 jsou parametry metody a g je gradient. Váhy se poté upraví podobně jako v metodě RMSprop viz rovnice (14), kde ϵ je malá hodnota pro prevenci dělení nulou a η je learning rate. Tato metoda je v dnešní době jedna z nejpoužívanějších a dosahuje nejlepších výsledků při trénování neuronových sítí[5].

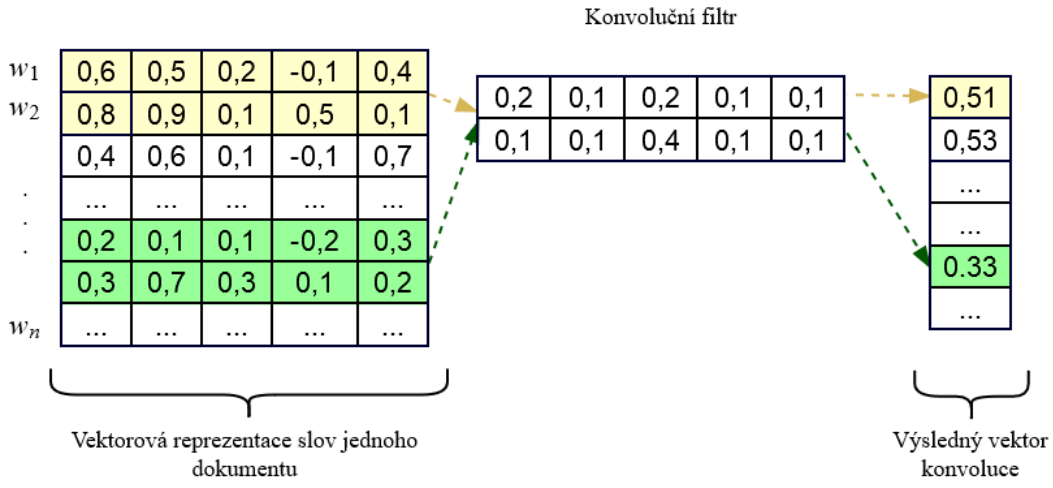
$$\theta_{t+1} = \theta_t - \frac{\eta}{\sqrt{\widehat{vt}} + \epsilon}\widehat{m_t} \tag{14}$$

[5] https://arxiv.org/abs/1412.6980

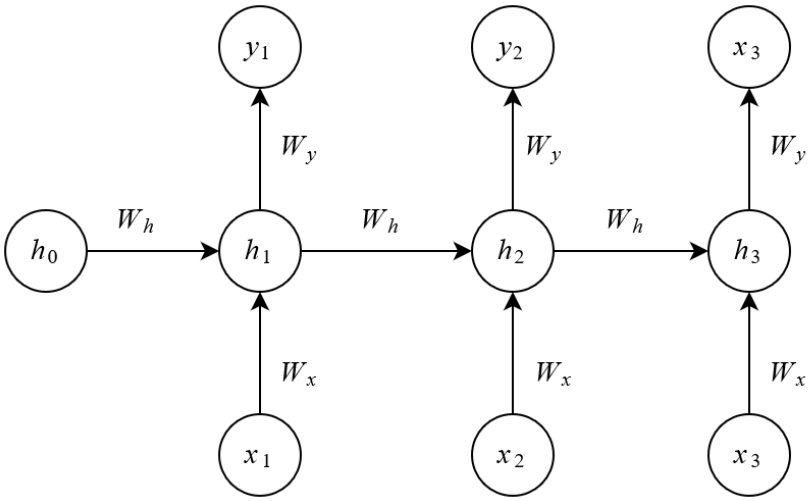
4.4.4 Konvoluční neuronové sítě (CNN)

Konvoluční sítě byly původně vymyšleny pro zpracování obrazu, později se ukázalo, že jsou velmi efektivní také v oblasti NLP a dosahují excelentních výsledků při úlohách jako sémantický rozbor, modelování vět a dalších tradičních úloh z oboru NLP [21]. Pokud řešíme úlohu, kde záleží na pořadí prvků v množině (např. pořadí slov ve větě nebo pořadí vět v dokumentu), jako třeba analýza sentimentu, konvoluční sítě mohou být velmi efektivní a robustní řešení. CNN obsahují jednu nebo více konvolučních vrstev a taky pooling vrstvy. Konvoluční vrstvy předzpracovávají vstupní data pomocí konvolučních filtrů, jinak řečeno hledají užitečné informace ve velké struktuře, a jejich kombinací tvoří vektory reprezentující danou strukturu a její důležité aspekty. Součástí sítě jsou i plně propojené vrstvy (angl. *fully connected layers*), které se typicky vyskytují na konci a provádí klasifikaci za pomocí informací z předchozích vrstev.

Hlavní myšlenkou konvolučních sítí v oboru NLP je aplikace nelineárních funkcí na každou instanci posuvného okna (angl. *sliding window*) o velikosti k (počet slov na které se funkce aplikuje), v dokumentu d (viz obr. 10) [21]. Tyto funkce neboli konvoluční filtry, transformují okno k slov do x-dimenzionálního vektoru, jež zachycuje důležité vlastnosti slov v posuvném okně. Při zpracování textu se používá 1D konvoluce, to znamená, že šířka konvolučního filtru má vždy stejnou velikost jako vstupní matice a proměnlivá je pouze výška posuvného okna, určující na kolik slov se bude filtr aplikovat (typicky 2-5 slov). Výstupní vektory se nazývají příznakové mapy (angl. *feature maps*). Po konvoluci se běžně provádí operace „pooling“, která kombinuje výsledné vektory vzniklé použitím různých filtrů a vytvoří nový redukovaný vektor. Redukce dimenze se provádí sloučením p prvků, výběrem maximální nebo průměrné hodnoty. Díky redukci vektorů lze efektivně zpracovávat mnohem větší dokumenty a extrahovat pouze nejdůležitější informace.



Obrázek 10: Znázornění konvoluce s filtrem pro 2 slova



Obrázek 11: Rozvinutá struktura rekurentní sítě

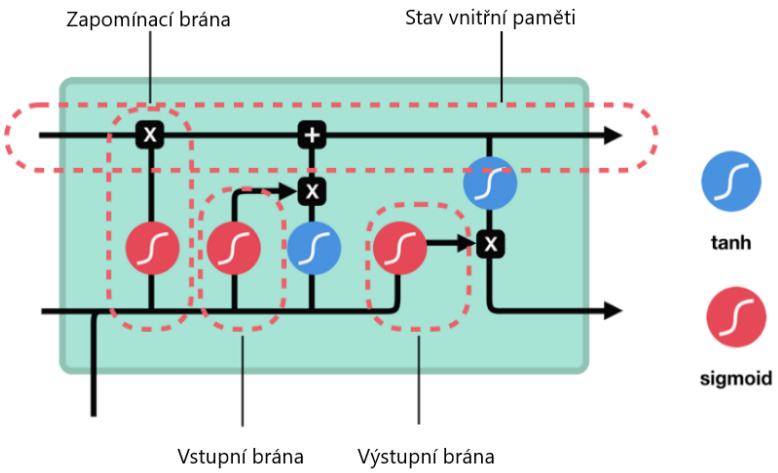
4.4.5 Rekurentní neuronové sítě (RNN)

V tradiční dopředné neuronové síti předpokládáme, že jsou všechny vstupy sítě na sobě nezávislé. Naopak rekurentní sítě jsou neuronové sítě s „vnitřní pamětí“, kde na každý vstup je aplikována stejná funkce, jejíž výstup není ovlivněn pouze váhou daného vstupu, ale také výstupem funkce v předchozím kroku (viz obr. 11). Tudíž dva stejné vstupy mohou mít rozdílný výstup v závislosti na předchozích vstupech. Tento vnitřní stav (paměť) reprezentuje kontext daného vstupu a je vhodný pro situace, kdy máme na vstupu sítě sekvence prvků, jež se navzájem ovlivňují a jejich vzájemná poloha je důležitá pro klasifikaci. RNN se používá například pro predikci časových řad, strojový překlad nebo generování textu.

Bidirectional RNN

Pokročilejší variantou jsou Bidirectional RNN neboli obousměrné rekurentní sítě. Rekurentní vrstva je rozdělena do dvou částí, první část zpracovává vstupy postupně od začátku a druhá část je zpracovává naopak od konce sekvence. Nakonec se výstupy obou částí spojí a přechází do další vrstvy. Díky tomu síť bere v úvahu předchozí prvky i následující prvky. Tato technika je velmi užitečná pro úkoly, kde správná predikce závisí i na budoucích vstupech nebo na celé sekvenci.

Původní rekurentní sítě s běžnými uzly se v praxi příliš nepoužívají, kvůli problému s mizejícím gradientem. Tento problém nastává, pokud chceme naučit síť závislosti mezi slovy nebo hodnotami sekvencí, které jsou odděleny větším množstvím jiných prvků. Důvodem je, že malý gradient nebo váhy (hodnoty menší než 1) jsou mnohokrát násobeny samy sebou a gradient se začne blížit asymptoticky k nule. To způsobí, že váhy se pro tyto spojení nijak znatelně nemění a síť se není schopna naučit závislosti více vzdálených prvků.



Obrázek 12: Struktura LSTM bloku (vlastní úprava, převzato z [22])

LSTM (Long Short Term Memory)

Aby bylo možné využívat hluboké rekurentní sítě v praxi, bylo potřeba vyřešit problém mizejícího gradientu. Jako jedno z řešení vznikly LSTM bloky (viz obr. 12), jež nahrazují klasické uzly RNN. Specificky navržené logické jednotky k zachování dlouhodobé informace, které dostatečně redukují tento problém. Uchovávají navíc stav vnitřní paměti, který je přidán ke zpracovávanému vstupu a tím je výrazně snížen multiplikační efekt malých gradientů. Přístup ke stavu vnitřní paměti je řízen konceptem bran, což jsou hladké matematické funkce, simulující logické brány. LSTM používá tři druhy bran: vstupní brány, výstupní brány a zapomínací brány (angl. *forget gates)*. Pro každý vstup se používají brány k rozhodnutí, kolik nových informací bude zapsáno do vnitřní paměti, kolik aktuálního obsahu paměti by mělo být zapomenuto (vymazáno) a jaké informace z vnitřní paměti mají být poslány na výstup bloku.

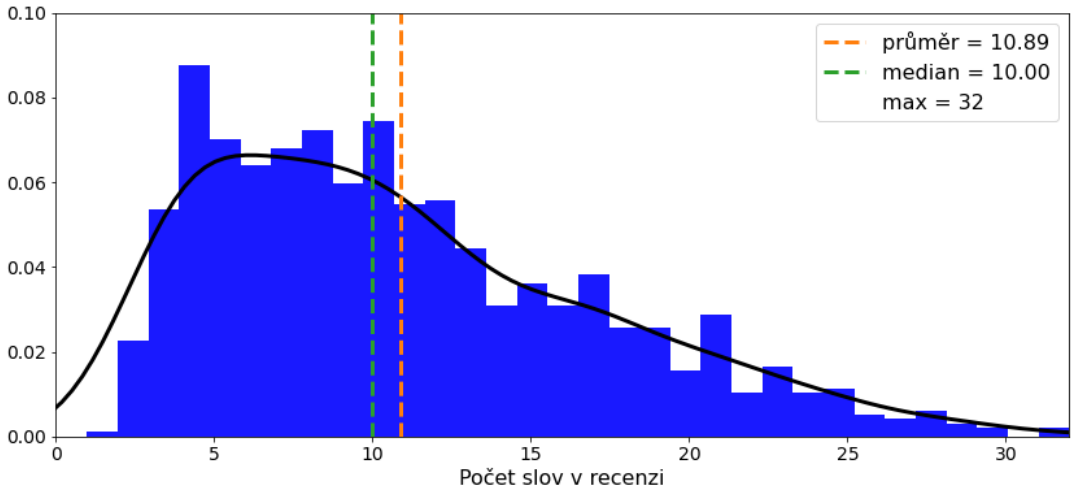
5 Implementace

Účel této kapitoly je popsání datových sad, na kterých boudou provedeny experimenty a porovnání algoritmů. Dále je zde uvedena implementace všech testovaných algoritmů, včetně technik předzpracování dat pro jednotlivé algoritmy. Tato kapitola taktéž popisuje návrh vlastních struktur neuronových sítí.

5.1 Datové sady

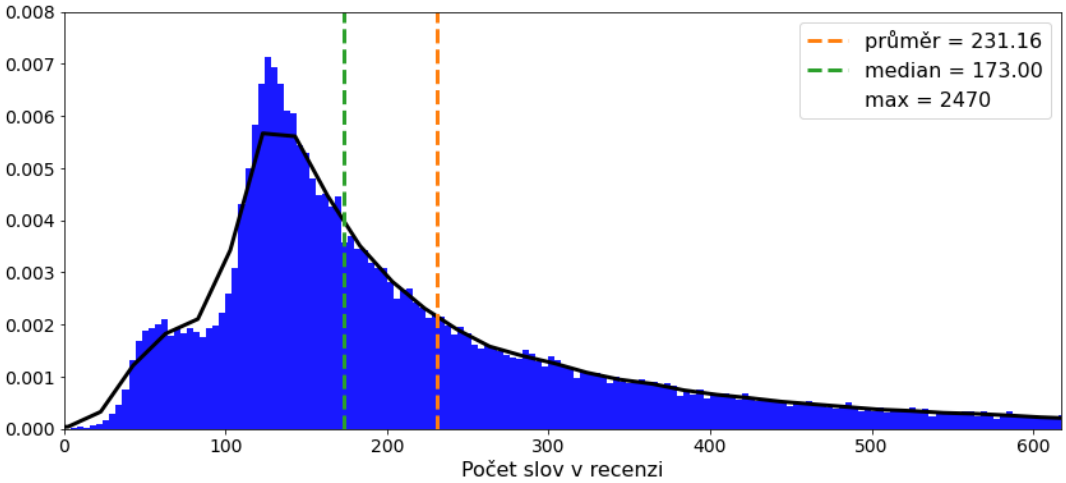
Pro účely testování vybraných metod byly vybrány tři datové sady, které jsou využívány v následujících experimentech. Datové sady byly vybrány tak, aby prověřily testované algoritmy s menším i větším počtem dat. Největší datová sada byla upravena s ohledem na dobu trvání provedení experimentů. Všechny datové sady obsahují recenze v anglickém jazyce.

První datová sada **Yelp** [23] obsahuje náhodně vybrané uživatelské recenze restaurací z webového portálu yelp.com, které byly následně doplněny o anotace vyjadřující sentiment. Obsahuje 500 pozitivních a 500 negativních jedno-větných recenzí, velikost textového souboru je 85 kB. Recenze mají dohromady 10894 slov, což je v průměru 11 slov na jednu recenzi. Recenze jsou vždy silně pozitivní nebo silně negativní. Tato datová sada je velmi malá a recenze se skládají z malého počtu slov, byla vybrána především k otestování, jak si algoritmy poradí s nedostatkem trénovacích dat.



Obrázek 13: Distribuce počtu slov recenzí v datové sadě Yelp

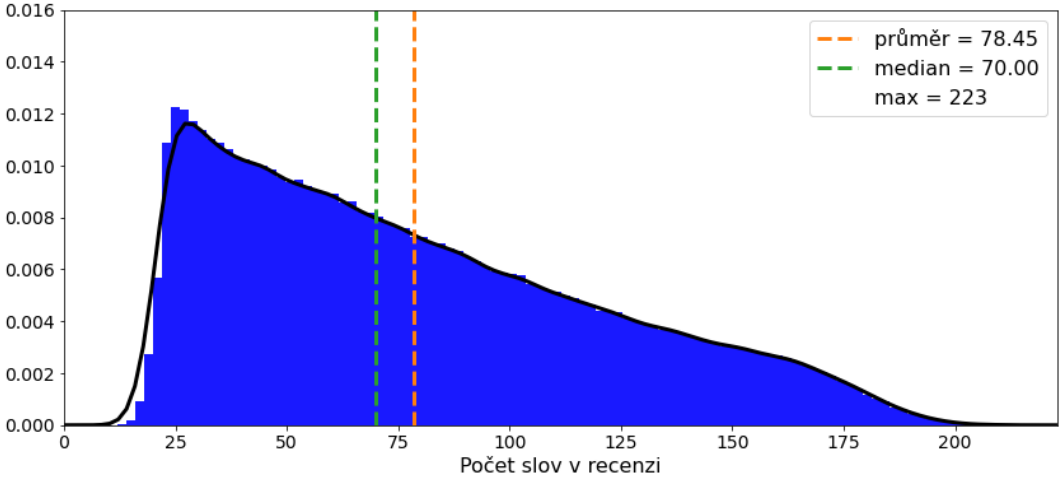
Druhá datová sada **IMDB** [24] se skládá z 50 000 filmových recenzí speciálně vybraných pro analýzu sentimentu na webovém portálu imdb.com. Klasifikace sentimentu byla provedena podle hodnocení filmu, kdy hodnocení menší než 5 znamená negativní klasifikaci a hodnocení 7 a více znamená pozitivní klasifikaci. K jednomu filmu se může vztahovat maximálně 30 recenzí. Recenze jsou výrazně delší než v první datové sadě, dohromady obsahují více než 11,5 milionů



Obrázek 14: Distribuce počtu slov recenzí v datové sadě IMDB

slov, průměrně 231 slov v jedné recenzi. Datová sada je vyvážená a obsahuje pozitivní a negativní recenze v poměru 1:1. Velikost textového souboru je 63 MB.

Největší použitou datovou sadu **Amazon**, vytvořil Xiang Zhang pro testování algoritmů ve své studii [25]. Obsahuje recenze různých produktů, které byly sbírány po dobu 18 let na webovém portálu Amazon.com. Sentiment byl odvozen z uděleného hodnocení autorem recenze. Hodnocení 1 a 2 bylo určeno jako negativní a hodnocení 4 a 5 jako pozitivní. Recenze s hodnocením 3 byly vyřazeny. Datová sada je rozdělená na 3,6 miliónu testovacích dat a 400 tisíc trénovacích dat. Pro účely této práce byla testovací datová sada zmenšena na 1 milion recenzí. Ze zbylých recenzí byla vytvořena ještě validační datová sada obsahující 100 tisíc recenzí. Recenze jsou kratší než v případě datové sady IMDB, průměrně se jedna recenze skládá ze 78 slov. Všechny části datové sady obsahují stejné množství negativních a pozitivních recenzí. Velikost textového souboru s trénovacími daty je 404 MB.



Obrázek 15: Distribuce počtu slov recenzí v datové sadě Amazon

5.2 ML .NET

Implementace metod SDCA a Averaged Perceptron byla provedena v jazyku C# (.NET Framework 4.7.2) pomocí frameworku ML .NET[6] (verze 1.4.0). Tento framework byl vytvořen společností Microsoft v roce 2019 a od té doby je stále vylepšován. Je multiplatformní a přináší do technologie .NET nativní podporu strojového učení pro produkční využití.

Datové sady nejsou načteny do paměti celé, ale jsou čteny po dávkách přímo z disku nebo databáze. To umožňuje zpracovávat datové sady obsahující miliony instancí bez omezení ze strany hardwaru. Aby tento princip fungoval, je potřeba vytvořit mapování atributů datové sady ze souboru na vlastní třídu, která reprezentuje jednotlivé instance (v tomto případě recenze a jejich anotace). V dalším kroku se definují metody pro předzpracování dat a následnou transformaci textu do vektorové reprezentace Bag of Words (viz kapitola 2.4.1). Metoda pro vektorizaci textu umožňuje specifikovat velikost n-gramů a metodiku pro hodnocení relevance slov. Všechny metody se řetězí do fronty v pořadí, ve kterém mají být vykonány. Na konec této fronty se přidá požadovaný klasifikátor a jeho parametry. Metoda SDCA umožňuje nastavit tyto parametry (v závorce za parametrem je uvedena hodnota ve výchozím nastavení):

- L1 a L2 regularizace dat (žádná regularizace se nepoužívá)
- Konstanta pro regulaci konvergence (0,1)
- Maximální počet iterací (neomezeně)

Druhá metoda Averaged Perceptron umožňuje nastavit:

- Loss funkci (Log loss)
- Learning rate (1)
- Počet iterací (1)
- L1 a L2 regularizace dat (žádná regularizace se nepoužívá)
- Postupné snižování parametru learning rate (neaktivováno)

Experiment v sekci 6.3 se zabývá hledáním optimálních parametrů a jejich detailnějším popisem. Když je definován celý proces od načtení dat až po klasifikátor, tak se spustí učení modelu. Jednotlivé dávky dat projdou celým procesem a poté se dávka zahodí a načte se do paměti další dávka z disku. U některých klasifikačních metod může být tento proces prováděn paralelně. Po dokončení trénování se provede evaluace modelu na testovacích datech. Následně je možné model uložit pro pozdější použití.

[6] https://dotnet.microsoft.com/apps/machinelearning-ai/ml-dotnet

5.3 FastText

FastText je open source knihovna, vyvinuta společností Facebook v roce 2016. Obsahuje moduly pro efektivní reprezentaci textu tvorbou word embeddings (viz kapitola 2.4.3) a klasifikaci textu pomocí lineárního modelu. Je napsaná v jazyce C++ s podporou paralelizace. Implementace metody a experimentů byla provedena v jazyku Python (verze 3.7). Oficiální aplikační rozhraní (API) pro python obsahuje pouze základní metody a neumožňuje složitější práci s daty, neboť umí zpracovat data pouze ze souboru se specifickou strukturou. Další problém oficiálního API je nedostatečné nastavení evaluace modelu. Z tohoto důvodu je pro práci s knihovnou FastText použito API Skift[7] (verze 0.0.17), jež umožňuje pracovat s daty pomocí knihoven Pandas a NumPy, a také zjednodušuje implantaci křížové validace nebo zvolení požadované metriky.

Pro práci s daty je použita knihovna Pandas, jež poskytuje struktury a nástroje pro organizaci dat a knihovna NumPy pro efektivní práci s vektory a maticemi. Předzpracování dat je více popsáno v experimentu 6.2. Po úpravě dat je potřeba vytvořit klasifikátor, jenž umožňuje mimo jiné natavit tyto parametry, které jsou dále popsány v kapitole 6.3.3:

- Velikost n-gramů
- Learning rate
- Počet epoch
- Loss funkce

5.4 Neuronové sítě

Obě neuronové sítě byly naimplementovány v programovacím jazyku Python (verze 3.7) pomocí frameworků TensorFlow (verze 2.2.0) a Keras. TensorFlow je aktuálně jedna z nejznámějších softwarových knihoven pro strojové učení primárně určena pro hluboké učení (angl. *deep learning*). Byla vyvinutá společností Google a zveřejněna v roku 2015. Umožňuje vytvářet strukturované grafy, jež se skládají z matematických operací a popisují, jak se data v grafu pohybují. Matematické operace jsou reprezentovány jako uzly grafu a spojení mezi nimi reprezentují vícerozměrné datové pole (tenzory). Složité matematické operace jsou implementovány převážně v jazyce C++, pro dosažení co nejlepší optimalizace. Pro používání poskytuje vysoko úrovňové API v jazyku Python a C++. Podporuje paralelizaci a zpracování pomocí CPU, GPU (CUDA) nebo TPU (Tensor procesing unit), funguje i na mobilních zařízeních.

Keras je open-source knihovna pro práci s neuronovými sítěmi v Pythonu. Keras byl navržen tak, aby byl uživatelský přívětivý, modulární, jednoduše rozšiřitelný. Knihovna sama poskytuje pouze aplikační rozhraní s vyšší abstrakcí, pro fungování potřebuje framework, který se stará o složité matematické operace v pozadí. Keras podporuje více frameworků nad kterými může

[7] https://github.com/shaypal5/skift

pracovat, mezi nejznámější patří TensorFlow, Theano, CNTK. V současné době je doporučován především TensorFlow a od verze 2.0 je Keras jeho součástí.

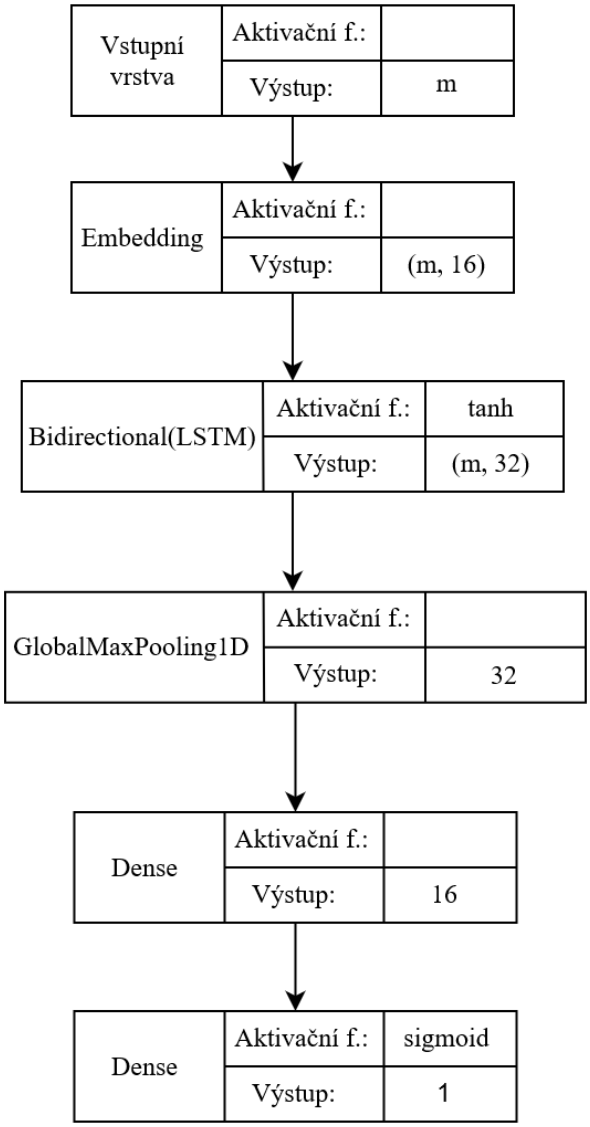
Pro práci s daty jsou opět použity knihovny Pandas a NumPy. Předzpracování dat je podrobně popsáno v sekci 6.2. Po předzpracování jsou jednotlivá slova nebo fráze převedeny do číselné reprezentace, kde každé číslo odkazuje na unikátní slovo v datové sadě. Počet unikátních slov byl omezen na 10000 nejčastějších, s výjimkou datové sady Yelp, která obsahuje pouze 2071 unikátních slov. Poté jsou všechny recenze zarovnány na stejnou délku, která byla vhodně zvolena pro každou datovou sadu podle distribuce délek recenzí. Pro datovou sadu Yelp to byla délka nejdelší recenze (32 slov). U IMDB byla délka omezena na 400 slov a v případě datové sady Amazon to bylo 180 slov. Kratší recenze byly doplněny nulami na požadovanou délku a delší recenze obsahovaly pouze daný počet slov.

Obě sítě využívají vektorovou reprezentaci slov v podobě word embeddings (viz kapitola 2.4.3). Tato reprezentace se nevytváří samostatně předem, ale tvoří se zároveň při trénování modelu. K tomu slouží Embedding vrstva, která se v sítí nachází hned za vstupní vrstvou, každý vstup převede do vektorové reprezentace a síť dále pracuje pouze s nově vzniklým vektorem. Vrstva umožňuje nastavit požadovanou dimenzi vektorového prostoru, která bude použita pro reprezentaci slov.

Další vrstva, kterou obsahují obě sítě se nazývá *GlobalMaxPooling1D* a slouží pro redukci dimenze. Redukce probíhá sloučením několika hodnot do jedné tak, že se vybere daný počet hodnot a ponechá se pouze ta nejvyšší. První část názvu vrstvy „Global“ znamená, že vybraný počet hodnot se rovná velikosti vstupu. Na vstupu vrstvy je matice $m \times n$ a na výstupu vznikne vektor o délce n.

5.4.1 LSTM

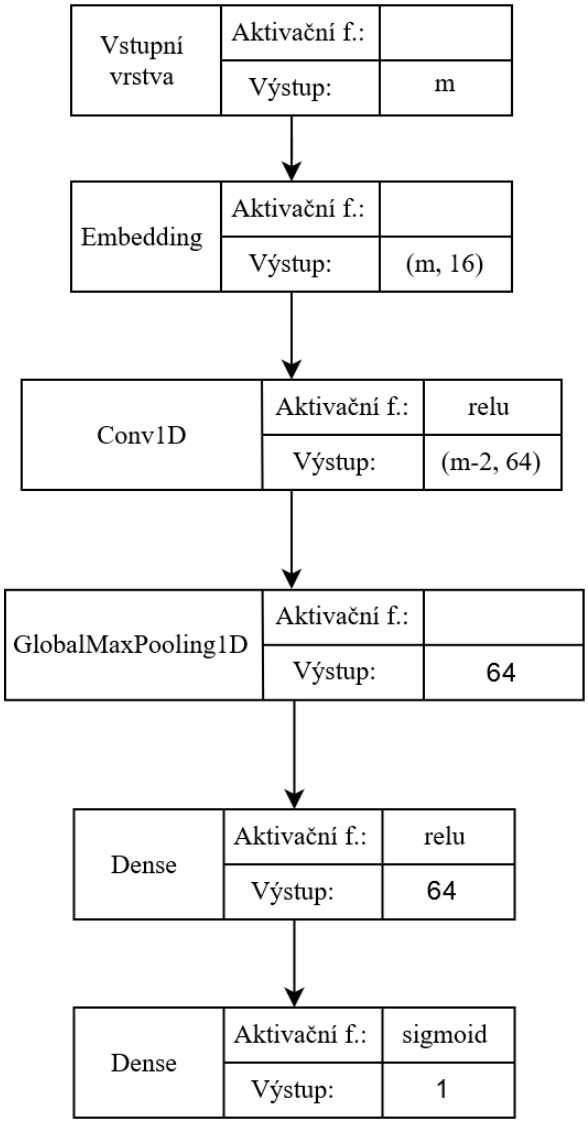
Navržená LSTM síť se skládá ze 6 vrstev, které jsou poskládány sekvenčně. Na obrázku 16 je znázorněna struktura sítě, kde m je maximální délka recenzí pro použitou datovou sadu. Vstupní vrstva pouze předává data do další vrstvy. V základní navržené konfiguraci Embedding vrstva tvoří word embeddings s 16 parametry pro každý term. Další vrstva je obousměrné LSTM, jenž umožňuje mimo jiné nastavit počet parametrů výstupního vektoru každé LSTM jednotky. Počet LSTM jednotek se odvíjí od rozměrů výstupní struktury předchozí vrstvy. V testované síti je zvoleno 16 parametrů, ale vrstva je obousměrná, tedy skládá se ze dvou vrstev, jejíchž výstup se spojí a výsledný výstup je matice o velikosti $m \times 32$ pro každou instanci (recenzi). Jako aktivační funkce je zvolena funkce Tanh. Poté následuje Global max pooling vrstva, jejíž výstup je předán Dense vrstvě neboli plně propojené vrstvě s 16 neurony. Na konci se nachází druhá Dense vrstva s jedním neuronem a aktivační funkcí Sigmoid, která slouží k výsledné binární klasifikaci.



Obrázek 16: Struktura navržené LSTM sítě

5.4.2 CNN

Struktura konvoluční sítě je představena na obrázku 17, konstanta m značí maximální zvolenou délku recenzí. Je opět složena z 6 vrstev, které jsou na sebe postupně napojeny. Vytvořené word embeddings mají 16 parametrů. Poté následuje 1D konvoluční vrstva s aktivační funkcí Relu. Je složena z 64 konvolučních filtrů a velikost kernelu je 3, což znamená, že filtr je v každém kroku aplikován na tři termy najednou. Od velikosti kernelu a počtu filtrů se odvíjí výstup vrstvy, který je reprezentován maticí o velikosti $(m-2) \times 64$. Čtvrtá vrstva je Global max pooling vrstva a za ní následují opět dvě plně propojené (Dense) vrstvy. První Dense vrstva má 64 neuronů a její aktivační funkce je Relu. Druhá Dense vrstva a zároveň poslední vrstva sítě má pouze jeden neuron a její aktivační funkce je Sigmoid.



Obrázek 17: Struktura navržené konvoluční sítě

Jako optimalizační algoritmus je u obou sítí použit optimalizátor Adam (viz kapitola 4.4.3). Aktuálně je to velmi populární algoritmus a dosahuje skvělých výsledků. Jelikož provádíme pouze binární klasifikaci, tak pro měření chybovosti predikce je použita loss funkce Binary cross-entropy. Trénování sítě probíhá iteračně ve 12 epochách, po každé epoše je provedena evaluace pomocí validačních dat. Pokud se model přestane zlepšovat, tak se trénování ukončí a všechny váhy jsou nastaveny na hodnoty z epochy, po které bylo dosaženo nejlepší evaluace. Trénování probíhá ve vhodně zvolených dávkách (angl. *batch*) podle datové sady.

6 Experimenty

Tato kapitola se zabývá porovnáním vybraných metod pro analýzu sentimentu v rámci textových dat. Následující kapitoly obsahují podrobně popsané experimenty a jejich výsledky. V poslední kapitole je souhrn nejlepších dosažených výsledků jednotlivých metod na všech datových sadách a jejich srovnání.

6.1 Základní porovnání

První experiment spočívá v porovnání úspěšnosti klasifikace jednotlivých metod v základním nastavení zvolené implementace. Pro potřeby klasifikačních algoritmů byl text převeden do numerické reprezentace na příznakové vektory (angl. *features vectors*) metodami s použitých knihoven. Metody Averaged Perceptron a SDCA používají vektorovou reprezentaci BoW. FastText a neuronové sítě vytváří vnoření slov (word embeddings). Úspěšnost byla měřená pomocí křížové validace rozdělením datové sady na 5 podmnožin o stejné velikosti. Jedna podmnožina dat vždy slouží jako testovací data, zatímco zbylé podmnožiny dat jsou použity pro natrénování modelu. Tento proces se opakuje vždy s jinou podmnožinou pro testovací data. Model se hodnotí procentuální úspěšností klasifikace na testovacích datech, tato metrika se nazývá přesnost modelu. Přesnost modelu ze všech 5 iterací byla zprůměrována a výsledky jednotlivých metod na obou datových sadách lze vidět v tabulce 1. Výjimkou byla datová sada Amazon, z důvodu velikosti by byla křížová validace výpočetně velmi náročná. Datová sada navíc obsahuje svoje testovací data s dostatečnou velikostí, a tudíž není potřeba dále dělit trénovací data. Evaluace modelu byla tedy provedena pouze jednou na těchto testovacích datech.

	Yelp Dataset	IMDB Dataset	Amazon Dataset
Averaged Perceptron	75,95 %	86,04 %	90,63 %
SDCA	**82,94 %**	84,56 %	87,86 %
FastText	58,30 %	87,90 %	90,85 %
LSTM	80,80 %	89,70 %	**94,12 %**
CNN	80,50 %	**89,72 %**	93,07 %

Tabulka 3: Přesnost modelů jednotlivých algoritmů na všech datových sadách

Na malé datové sadě (1000 prvků) si nejlépe vedla metoda SDCA s úspěšností klasifikace 82,24 %. Tato metoda je navržena tak, aby podávala dobré výsledky i při menší velikosti dat, zatímco ostatní metody jsou navrženy spíše pro větší datové sady. Metoda Averaged Perceptron dosáhla úspěšnosti 75,95 %, ale v základním nastavení dělá pouze jednu iteraci učení, což ji dost znevýhodňuje. Poměrně dobrý výsledek měly i obě neuronové sítě, jež se dostaly přes 80 %. Nejhůře dopadla metoda FastText, která je určena spíše pro datové sady s miliony slov.

Na střední datové sadě (50 000 prvků) dosáhly nejlepšího výsledku neuronové sítě, konvoluční neuronová síť měla o 2 setiny procenta přesnější klasifikaci a to 89,72 %. Nelze ale říct, že CNN

je nejlepší, rozdíl výsledků obou sítí je tak malý, že kvůli náhodným veličinám by se výsledky mohly při opětovném měření otočit. Obě neuronové sítě jsou navrženy pro tuto konkrétní úlohu a mají výhodu nad ostatními algoritmy v původní konfiguraci. Velmi dobře dopadla také metoda FastText s vysokou přesností 87,9 %. O něco hůře dopadla metoda Averaged Perceptron, která měla o 1,8 % horší výsledek, na druhou stranu má tato metoda v základu nastavenu pouze 1 iteraci, zatímco FastText provádí 5 iterací trénování. Metoda SDCA měla podobnou úspěšnost jako na menší datové sadě, konkrétně 84,56 %.

S největší datovou sadou Amazon (1,4 milionu prvků) byly opět velmi úspěšné neuronové sítě, zde si ale daleko lépe vedla LSTM síť s přesností modelu 94,12 %. CNN měla nižší přesnost o více než jedno procento. Metody FastText a Averaged Perceptron měly velmi podobné výsledky s přesností něco málo přes 90,50 %. Nejhůře se umístila metoda SDCA s velkým propadem téměř 3 % za metodou AP.

6.2 Předzpracování textu

Tento experiment se zabývá vlivem předzpracování textu na úspěšnost klasifikace. Pro předzpracování textu byly použity techniky pro odstranění stop slov, Lemmatizaci, odstranění interpunkce a také rozdělení textu na n-gramy různých velikostí. Přesnost modelu byla opět měřena křížovou validací rozdělením dat na 5 částí s výjimkou datové sady Amazon. Některé knihovny automaticky převádí text na malá písmena, pro účely porovnání byly proto všechny recenze převedeny na malá písmena.

Odstranění stop slov a interpunkce pro metody SDCA a Averaged Perceptron bylo provedeno pomocí knihovny ML .NET[8]. Tato knihovna nabízí vlastní slovníky stop slov pro různé jazyky (aktuálně obsahuje 16 jazyků) včetně češtiny, ale také umožňuje použití svého slovníku. V tomto experimentu byl použit nabízený slovník pro jazyk angličtina. V základním nastavení je interpunkce ponechána a při tokenizaci je součástí slov. Pro metodu FastText a neuronové sítě byla interpunkce odstraněna regulárním výrazem v jazyce Python. Odstranění stop slov bylo provedeno podle slovníku anglických stop slov z platformy NLTK[9].

Pro účely experimentu byla pro lemmatizaci zvolena open source knihovna LemmaGen[10]. Tato knihovna podporuje 12 různých jazyků (včetně češtiny) a nabízí implementaci pro programovací jazyky c++, C# a python. Lemmatizace nezávisí na struktuře textu, text je jako první tokenizován na jednotlivá slova a poté jsou tato slova lemmatizována podle rozsáhlého slovníku. Společně s lemmatizací slov je odstraněna i interpunkce, tudíž nelze porovnat vliv interpunkce na kvalitu klasifikace. Pomocí této knihovny byly vytvořeny nové datové sady, obsahující recenze po lemmatizaci. Nově vzniklé datové sady byly použity pro všechny metody v následujícím experimentu.

[8] https://dotnet.microsoft.com/apps/machinelearning-ai/ml-dotnet
[9] https://www.nltk.org/
[10] http://lemmatise.ijs.si/

Rozdělení textu na n-gramy (množiny n po sobě jdoucích slov) bylo opět provedeno knihovnou ML.NET. U všech datových sad byly testovány n-gramy o délce 1 až 3 slov. Zároveň se všemi velikostmi n-gramů byly testovány jiné metodiky hodnocení relevance slov. V základním nastavení knihovna ML.NET používá metodiku TF, další použité metodiky jsou IDF a TF-IDF (viz kapitola 2.4.2). FastText vytváří příznakové vektory pomocí predikce, tudíž nelze použít metodiky jako TF-IDF. Proto jsou testovány pouze různé velikosti n-gramů. N-gramy byly vytvořeny knihovnou FastText.

U neuronových sítí byly jako n-gramy tvořeny vhodné kolokace (slovní spojení). Kolokace jsou spojení slov, která spolu souvisejí gramaticky i sémanticky a vytvářejí víceslovné pojmenování. Spojování slov bylo provedeno na základě studie T. Mikolova [26] za pomocí knihovny Gensim[11]. Pro každé dva termy w_i, w_j se vypočte skóre pomocí rovnice (15), kde δ je parametr, jež zabraňuje vzniku velkého množství frází z málo četných termů. Termy jsou spojeny pouze v případě, že výsledné skóre je vyšší než konstanta *threshold*. Tímto způsobem je možné vytvářet užitečné fráze bez masivního zvětšení abecedy unikátních slov, jak je tomu při tvorbě kompletních n-gramů.

$$skóre(w_i, w_j) = \frac{počet(w_i, w_j) - \delta}{počet(w_i) \cdot počet(w_j)} \tag{15}$$

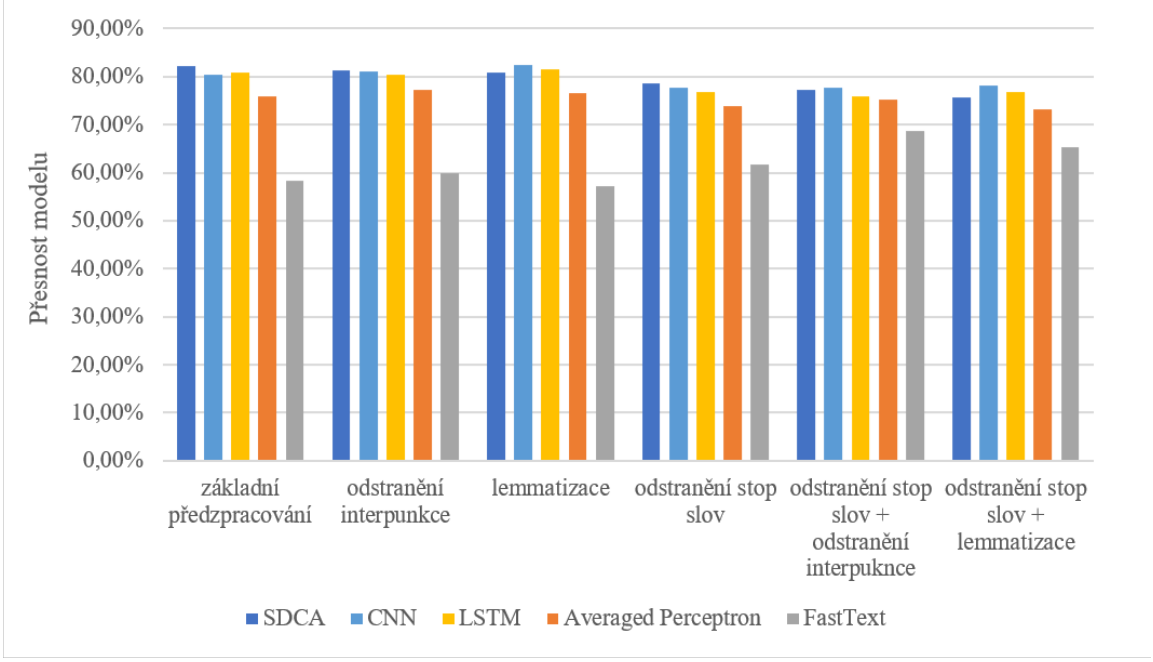
6.2.1 Datová sada Yelp (1000 recenzí)

U malé datové sady mělo předzpracovaní textu většinou neblahý vliv na přesnost klasifikace, především u metody SDCA. S touto metodou všechny techniky předzpracovaní textu snížily přesnost klasifikace. Zatímco lemmatizace a odstranění interpunkce znamenalo snížení přesnosti o 1 až 1,5 %, při odstranění stop slov už to bylo snížení přesnosti téměř o 4 %, kombinace těchto technik dále snižovala přesnost (viz obr. 18). Datová sada je malá a tyto techniky ji ještě více zmenší, což zřejmě způsobuje, že žádná technika předzpracování nepřinesla zlepšení. Kompletní výsledky jsou v tabulce 26 (příloha A).

S metodou Averaged Perceptron mělo předzpracovaní textu o něco lepší výsledky. Lemmatizace zvýšila přesnost klasifikace o půl procenta a odstranění interpunkce zlepšilo přesnost o 1,35 %. Ostatní metody předzpracování nebo jejich kombinace, přesnost klasifikace vždy zhoršily. Výsledky s použitím metody FastText ukázaly jedno výrazné zlepšení, a to při odstranění stop slov a interpunkce, přesto byla přesnost stále znatelně horší než u ostatních metod.

U obou neuronových sítí znatelně pomohla lemmatizace textu, v případě CNN bylo dosaženo dokonce nejlepších výsledků z testovaných metod a to 82,40 %, čímž nepatrně překonala metodu SDCA. Odstranění interpunkce mírně zvýšilo přesnost o 0,60 % oproti základnímu předzpracování. U LSTM byla lemmatizace jediná technika, která zvýšila přesnost klasifikace. S použitím této techniky model dosáhl přesnosti 81,50 %.

[11] https://radimrehurek.com/gensim/



Obrázek 18: Přesnost modelu při všech technikách předzpracování textu na datové sadě Yelp

Další testovaná technika předzpracování textu se týká rozdělení textu na n-gramy pro tvorbu příznakových vektorů a zároveň vyzkoušením jiných metodik hodnocení relevance těchto n-gramů. Pro každou hodnotu n u tvořených n–gramů, jsou testovány tři metodiky a to TF, IDF a TF-IDF. Z předchozích technik předzpracování textu bylo pro metodu SDCA zvoleno základní předzpracování z důvodu nejlepších výsledků. V případě metody Averaged Perceptron byly testy provedeny se základním předzpracováním a také s odstraněním interpunkce. U metody FastText bylo provedeno odstranění stop slov a interpunkce. V případě neuronových sítí byla použita lemmatizovaná verze datové sady pro obě sítě.

n	TF	IDF	TF-IDF
1	82,24	82,61	**83,32**
2	79,58	79,81	79,91
3	80,48	80,39	80,39

Tabulka 4: Přesnost modelu metody SDCA pro kombinace n-gramů a metodik hodnocení relevance n-gramů na datové sadě Yelp (v procentech)

Jak můžeme vidět v tabulce 4, s metodou SDCA byla dosažena nejvyšší přesnost modelu 83,32 % kombinací n-gramů o velikosti 1 a metodiky TF-IDF. O trochu hůř dopadla metodika IDF se stejnými n-gramy. Při tvorbě n-gramů o velikosti dva dosahovala metodika TF nejhorších výsledků až 79,58 %. Metodiky IDF a TF-IDF měly s většími n-gramy velmi podobné nebo stejné výsledky, což je dáno tím, že jednotlivé n-gramy se začaly vyskytovat pouze jednou v celé datové sadě.

n	TF	IDF	TF-IDF
1	75,95	**78,87**	78,41
2	76,3	77,09	76,7
3	75,27	75,76	75,76

(a) Základní předzdpracování

n	TF	IDF	TF-IDF
1	77,3	**78,45**	78,02
2	76,55	76,1	76,31
3	73,68	73,74	73,74

(b) S odstraněním stop slov

Tabulka 5: Přesnost modelu metody Averaged Perceptron pro kombinace n-gramů a metodik hodnocení relevance n-gramů na datové sadě Yelp (v procentech)

V tabulkách 5a a 5b můžeme porovnat výsledky pro klasifikační metodu Averaged Perceptron. V obou případech nejlépe dopadla metodika IDF s velikostí n-gramů 1, o něco lepší byla se základním předzpracováním, kde vytrénovaný model měl přesnost 78,87 %. Se zvyšující se velikostí n-gramů přesnost velmi klesala. Po odstranění interpunkce metodika IDF překvapivě nepřinesla takové zlepšení, jako v případě základního nastavení. Za zmínku stojí jediné zlepšení s využitím větších n-gramů, a to v případě metodiky TF a základního předzpracování dat, kde bylo dosáhnuto použitím n-gramů o velikosti dva zlepšení o 0,35 %. V tabulce 6 můžeme vidět, že stejně jako v předchozích případech, u metody FastText bylo dosaženo nejlepších výsledků s uni-gramy. Bi-gramy i tri-gramy značně snížily přesnost modelu.

n	Přesnost
1	68,70
2	61,90
3	62,90

Tabulka 6: Přesnost modelu metody FastText pro různé velikosti n-gramů na datové sadě Yelp (v procentech)

V případě neuronových sítí byly spojovány slova do frází (kolokací). Ačkoliv bylo ve vykonaném experimentu zaznamenáno mírné zlepšení s použitím frází (viz tabulky 7a a 7b), tak výsledky obou sítí byly velmi náhodné, a tudíž nelze jednoznačně určit jaká kombinace parametrů je nejvhodnější.

	10	20	40
1	81,50	81,50	81,50
2	80,00	81,80	80,20
3	81,60	80,10	80,30

(a) LSTM

	10	20	40
1	82,40	82,40	82,40
2	82,90	81,30	82,70
3	82,30	82,80	82,20

(b) CNN

Tabulka 7: Přesnost modelů neuronových sítí s frázemi o velikosti 1–3 a různými hodnotami parametru threshold, datová sada Yelp (v procentech)

6.2.2 Datová sada IMDB (50 000 recenzí)

Na větší datové sadě už mělo předzpracování textu daleko lepší vliv. Veškeré techniky předzpracování textu měly pozitivní vliv na přesnost klasifikačních metod (viz graf 19) s výjimkou neuronových sítí. U metod SDCA a Averaged Perceptron měly tyto techniky podobný účinek. Pro obě metody nejlépe dopadlo trénování modelu po odstranění stop slov v kombinaci s odstraněním interpunkčních znamének. Metoda Averaged Perceptron dosáhla přesnosti modelu 87,77 % a Metoda SDCA 86,60 %. Samostatně měla největší vliv technika odstranění stop slov a nejhůře dopadla lemmatizace, avšak stále s lepšími výsledky než jen se základním předzpracováním textu (transformace na malá písmena, odstranění diakritiky, vektorizace textu).

	Se všemi slovy	Bez stop slov
Bez lemmatizace	11,55 M	6,36 M
S lemmatizací	11,68 M	6,32 M

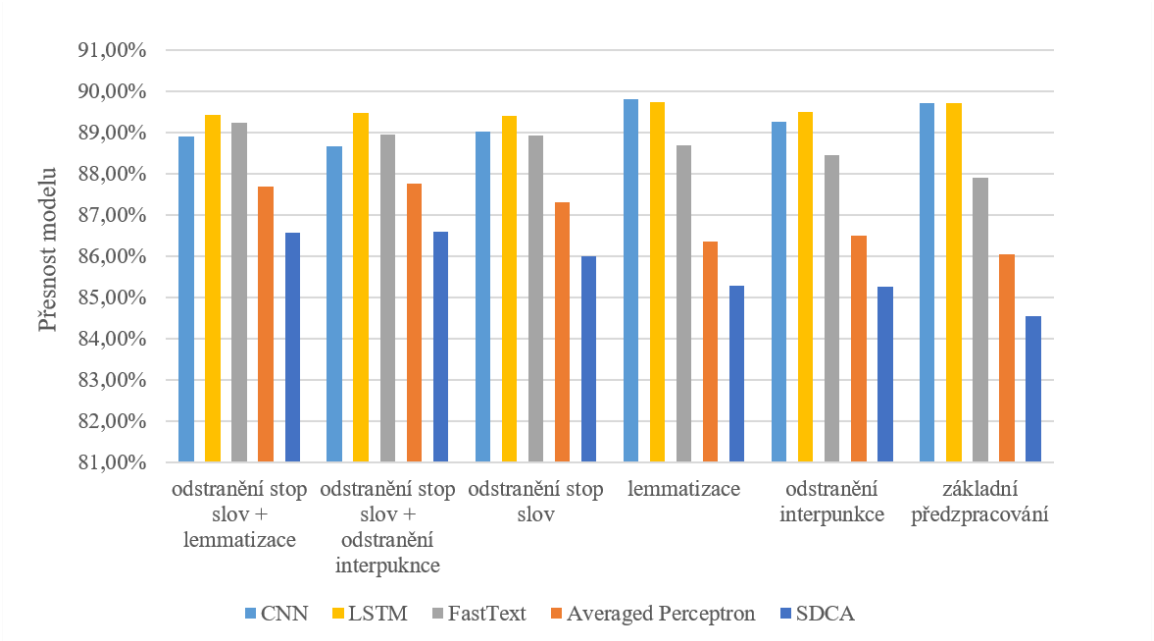
Tabulka 8: Počty slov v datové sadě IMDB po použití technik předzpracování textu (v milionech)

S použitím metody FastText, lemmatizací datové sady a odstraněním stop slov bylo dosáhnuto skvělého výsledku 89,23 %. Díky předzpracování textu jsme tedy zlepšili přesnost modelu o 1,33 % i přesto, že odstraněním stop slov bylo z datové sady odstraněno spoustu záporů. Další výhodou této kombinace technik (lemmatizace, odstranění stop slov) je znatelné snížení výpočetní náročnosti učení modelu. Poněvadž se v angličtině často používají zkrácené tvary, jejichž lemmatizací vzniknou dvě slova, tak samostatnou lemmatizací se celkový počet slov zvýšil. Následně se ale celkový počet slov snížil odstraněním stop slov z lemmatizované datové sady o necelých 46 % (viz tabulka 8).

U neuronových sítí mělo překvapivě předzpracování textu spíše negativní efekt. Pouze v případě lemmatizace došlo k nepatrnému zlepšení u LSTM sítě o 0,04 % a o 0,08 % v případě CNN. Všechny ostatní techniky znatelně snížili přesnost klasifikace. Přesto konvoluční síť dosáhla nejlepších výsledků celkově, s lemmatizovanou datovou sadou byla přesnost modelu 89,80 %. Kompletní výsledky jsou v tabulce 27 (příloha A).

Druhá část experimentu se zabývá spojováním termů. Jako první byla testována metoda SDCA, kde rozdělení textu na n-gramy s délkou vetší než 1 zvýšilo přesnost modelu pouze v jednom případě a to při n-gramech o délce 2 se základní metodikou TF (viz tabulka 9a). Změna metodiky pro hodnocení n-gramů při základním předzpracování textu i při kombinaci technik pro odstranění interpunkce a stop slov měla v obou případech lepší výsledky. Především s metodikou IDF a pokročilejší úpravou textu model dosáhl nejvyšší přesnosti 87,16 %, což je zvýšení přesnosti o 0,56 %.

V tabulkách 10a a 10b můžeme vidět výsledky pro metodu Averaged Perceptron. Díky změně metodiky hodnocení n-gramů dosáhla výrazného zlepšení přesnosti modelu. Obě alternativní metodiky mají lepší výsledky než metodika TF a mezi nimi už jsou jen drobné rozdíly. V obou pří-



Obrázek 19: Přesnost modelu při všech technikách předzpracování textu na datové sadě IMDB

n	TF	IDF	TF-IDF
1	84,56	**86,40**	86,36
2	84,74	83,35	83,57
3	83,29	82,75	82,75

(a) Základní předzdpracování

n	TF	IDF	TF-IDF
1	86,6	**87,16**	86,84
2	84,07	83,43	83,49
3	83,35	83,28	83,28

(b) Pokročilé předzdpracování

Tabulka 9: Přesnost modelu metody SDCA pro kombinace n-gramů a metodik pro ohodnocení n-gramů na datové sadě IMDB (v procentech)

padech předzpracování textu je nejlepší metodikou IDF. Při základním předzpracování textu to je s rozdílem 2,1 % oproti TF. Při pokročilejším předzpracovaní textu (odstranění interpunkce a stop slov) je rozdíl oproti metodice TF 0,79 % a nejlepší dosažený výsledek je 88,56 %. Ačkoliv jsme dosáhli nejlepších výsledků s použitím technik pro odstranění interpunkce a stop slov, tak u datové sady se základním předzpracováním bylo zlepšení výrazně vyšší.

FastText byla jediná metoda, u které mělo rozdělení textu na větší části než uni-gramy pozitivní vliv na přesnost modelu (viz tabulka 11). Použitím bi-gramů bylo dosaženo zvýšení přesnosti na 89,48 %. Rozdělení textu na tri-gramy naopak mělo nepříznivý účinek a přesnost modelu se snížila o 0,82 % vůči uni-gramům.

Tvorba kolokací nepřinesla zlepšení přesnosti ani u větší datové sady (viz tabulky 12a a 12b). Obě sítě měly stále lepší výsledek bez použití frází. S parametrem $n = 1$ se kolokace netvoří, proto jsou hodnoty stejné pro všechny hodnoty parametru *threshold*. Čím je tento parametr menší, tím více termů je spojeno. U konvoluční sítě můžeme pozorovat zvyšování přesnosti

n	TF	IDF	TF-IDF
1	86,04	**88,14**	88,12
2	86,83	86,19	86,29
3	85,54	84,86	84,91

(a) Základní předzdpracování

n	TF	IDF	TF-IDF
1	87,77	**88,56**	88,42
2	86,83	86,19	86,29
3	85,54	84,86	84,91

(b) Pokročilé předzdpracování

Tabulka 10: Přesnost modelu metody AP pro kombinace n-gramů a metodik pro hodnocení n-gramů na datové sadě IMDB (v procentech)

n	Přesnost
1	89,23
2	**89,48**
3	88,41

Tabulka 11: Přesnost modelu metody FastText pro různé velikosti n-gramů na datové sadě IMDB (v procentech)

modelu s rostoucím parametrem *threshold* (méně frází). V případě LSTM vypadají výsledky spíše náhodné, ačkoliv je zajímavé, že s hodnotami *threshold* 10 a 40 byl výsledek stejný.

	10	20	40
1	**89,74**	**89,74**	**89,74**
2	89,58	89,41	89,58
3	89,49	89,70	89,48

(a) LSTM

	10	20	40
1	**89,80**	**89,80**	**89,80**
2	89,29	89,44	89,53
3	89,30	89,37	89,50

(b) CNN

Tabulka 12: Přesnost modelů neuronových sítí s frázemi o velikosti 1–3 a různými hodnotami parametru threshold, datová sada IMDB (v procentech)

6.2.3 Datová sada Amazon (1,4 M recenzí)

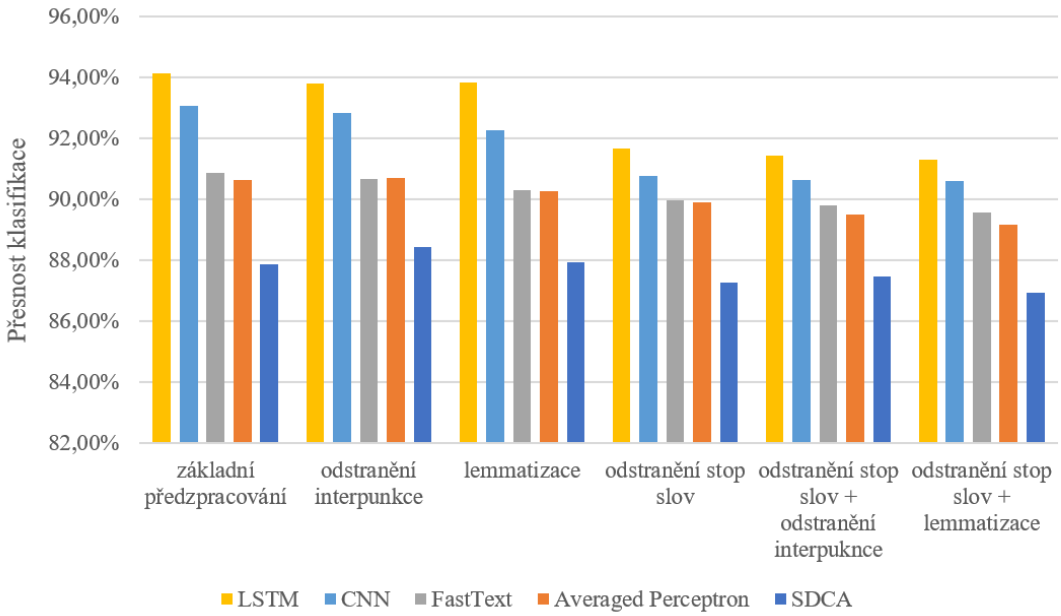
Testy na největší datové sadě Amazon ukázaly, že předzpracování textu mělo většinou negativní efekt (viz graf 20). Zlepšení bylo zaznamenáno pouze u metod SDCA a Averaged perceptor. V prvním případě bylo dosaženo zvýšení přesnosti o 0,59 % odstraněním interpunkce a nepatrné zlepšení použitím lemmatizace. Přesto měla metoda SDCA značně horší výsledky než ostatní metody. U druhé zmíněné metody byl pouze zanedbatelný nárůst díky odstranění interpunkce. Ačkoliv byla datová sada odstraněním stop slov zredukována téměř o 47 % (viz tabulka 13), pokles přesnosti byl příliš velký a využívat tuto techniku by nebylo dostatečně výhodné.

U metody FastText a neuronových sítí nebylo dosaženo žádného zlepšení, především použití techniky pro odstranění stop slov mělo za následek výrazné zhoršení klasifikace. Jedna recenze

	Se všemi slovy	Bez stop slov
Bez lemmatizace	78,45 M	41,82 M
S lemmatizací	79,27 M	41,41 M

Tabulka 13: Počty slov v datové sadě Amazon před a po použití technik předzpracování textu (v milionech)

obsahuje průměrně 78 slov a po odstranění stop slov se tato hodnota snížila na 41 slov, což může znamenat ztrátu významných informací pro klasifikaci sentimentu. Další důležitá informace pro určení sentimentu jsou zápory (v angličtině běžně vyjádřeny slovy „not“ nebo „no“), které jsou řazeny mezi velmi častá slova a při odstraňování stop slov jsou taktéž odstraněny. Kompletní výsledky jsou v tabulce 28 (příloha A).



Obrázek 20: Přesnost modelu při všech technikách předzpracování textu na datové sadě Amazon

Dále byly provedeny testy s tvořením n-gramů a v případě neuronových sítí s vyhledáním kolokací. Pro testování byly vybrány nejlepší techniky předzpracování textu z předchozího experimentu. První testovaná metoda byla SDCA, podobně jako u předešlé datové sady i zde byl mírný nárust přesnosti s využitím n-gramů o velikosti dvou termů. Přesto byl nejlepší výsledek 89,28 % dosažen s unigramy a změnou metodiky na IDF. U metody Averaged Perceptron je situace opačná, změna metodiky na TF zvýšila přesnost na 91,21 %, ale použití bigramů mělo za následek výrazné zlepšení o téměř 2 %. Jako nejlepší metodika se prokázala TF neboli četnost termů s přesností 92,54 %. Nicméně rozdíly mezi všemi metodikami byly zanedbatelné. Kompletní výsledky obou metod jsou v tabulkách 14a a 14b.

	TF	IDF	TF-IDF
1	88,45	**89,25**	89,24
2	88,81	87,82	87,89
3	87,48	86,68	86,73

(a) SDCA

	TF	IDF	TF-IDF
1	90,69	91,21	91,24
2	**92,54**	92,50	92,53
3	91,84	91,48	91,5

(b) Averaged Perceptron

Tabulka 14: Přesnost modelů metod SDCA a AP pro kombinace n-gramů a metodik hodnocení relevance n-gramů na datové sadě Amazon (v procentech)

	Přesnost
1	90,85
2	93,16
3	**93,28**

Tabulka 15: Přesnost modelu metody FastText pro různé velikosti n-gramů na datové sadě Amazon (v procentech)

Za zmínku stojí, že metoda Averaged Perceptron poprvé prokázala zlepšení přesnosti s použitím trigramů. To značí, že je tato datová sada vhodná pro techniky, které zachycují kontext dokumentu. Metoda FastText je založena na zaznamenání kontextu termů a jak lze vidět v tabulce 15, z větších n-gramů profitovala. Bigramy zvětšily přesnost o 2,31 % a trigramy dokonce o 2,43 %, čímž metoda FastText překonala konvoluční síť.

Ani u největší datové sady neměla tvorba kolokací pozitivní vliv na přesnost modelů neuronových sítí. Výsledky klasifikace obou sítí se zhoršují s rostoucím počtem nalezených frází (viz tabulky 16a a 16b). LSTM i konvoluční sítě samy o sobě zachytávají kontext dokumentu a spojováním termů do frází se mohou ztratit důležité informace. I přesto, že žádná technika předzpracování textu u datové sady Amazon nebyla pro LSTM prospěšná, tato síť dosáhla celkově nejvyšší přesnost klasifikace 94,12 %.

	10	20	40
1	**94,12**	**94,12**	**94,12**
2	93,82	94,08	94,05
3	93,80	93,84	93,87

(a) LSTM

	10	20	40
1	**93,07**	**93,07**	**9307**
2	92,88	92,93	93,01
3	92,72	92,77	92,71

(b) CNN

Tabulka 16: Přesnost modelů neuronových sítí s frázemi o velikosti 1–3 a různými hodnotami parametru threshold, datová sada Amazon (v procentech)

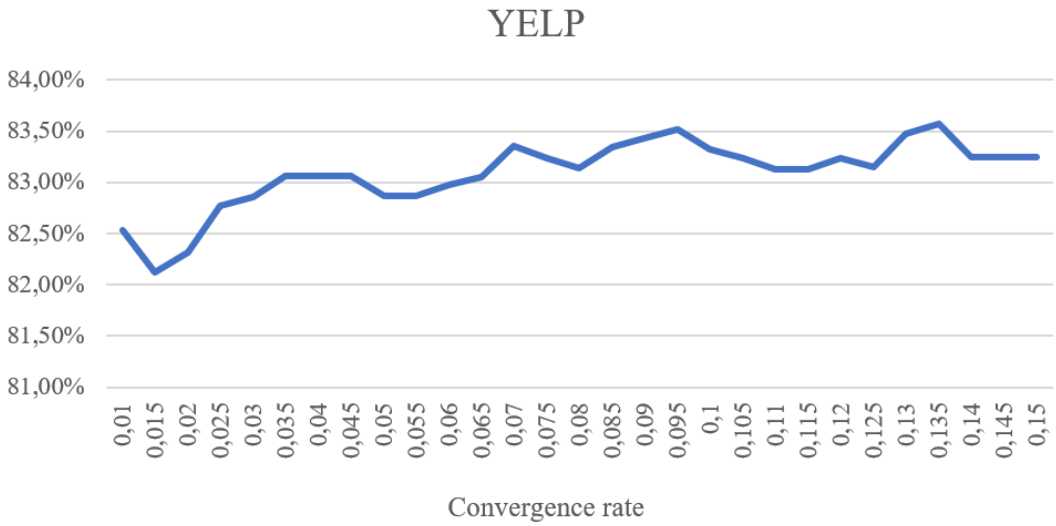
6.3 Úprava parametrů klasifikačních metod

Cílem tohoto experimentu je různá obměna parametrů klasifikačních metod tak, aby byla dosažena co nejvyšší přesnost modelu. V případě neuronových sítí je měněna architektura sítě. Způsob předzpracování datových sad pro jednotlivé metody je zvolen podle dosažených výsledků z předchozích experimentů.

6.3.1 SDCA

U metody SDCA hledáme optimální parametr *convergence rate*. Snižováním tohoto parametru zpomalujeme konvergenci, ale zvyšujeme riziko uváznutí v lokálním minimu. Nižší konvergence automaticky vede k více trénovacím iteracím. Hodnoty parametru musí být kladné a nesmí být větší než 1. V základním nastavení má tento parametr hodnotu 0,1. Pro nalezení optimální hodnoty je parametr testován v intervalu $\langle 0,01; 0,15\rangle$ s krokem 0,005. Přesný počet iterací nastavit nelze, je možné pouze omezit maximální počet, ale pro účely experimentu je zvolen neomezený počet iterací.

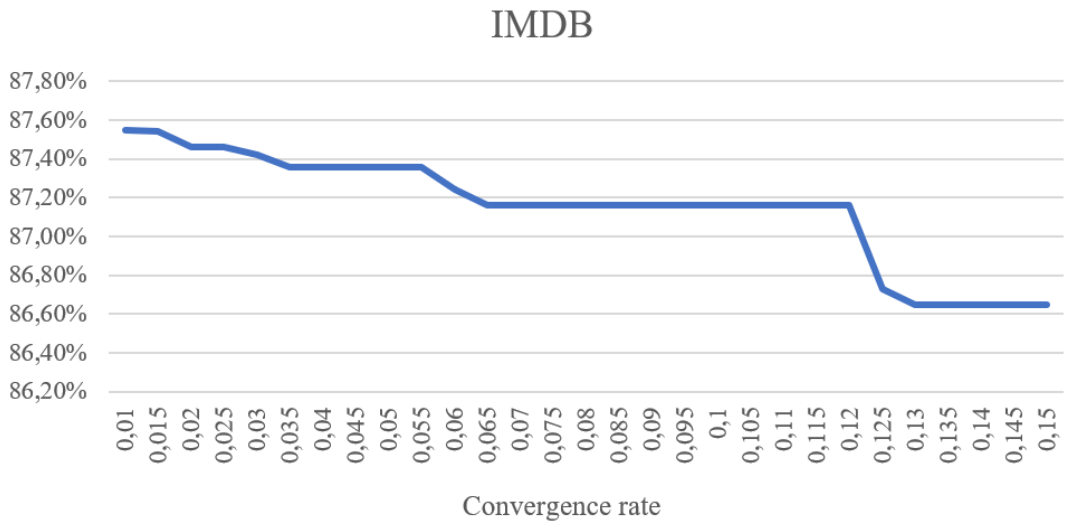
Na datové sadě Yelp byly změny velmi nepravidelné a snižováním tolerance konvergence přesnost modelu spíše klesala (viz obr. 21). Ačkoliv zvyšování tohoto parametru bylo také velmi kolísavé, tak nejlepšího výsledku bylo dosaženo při hodnotě 0,135 s přesností modelu 83,57 %, což je mírný narůst 0,25 %.



Obrázek 21: Přesnost modelu při různých hodnotách parametru „convergence rate“ u metody SDCA na datové sadě Yelp

Jak můžeme vidět na obrázku 22, na datové sadě IMDB se snižováním parametru *convergence rate* přesnost modelu postupně zlepšovala až do nejnižší testované hodnoty 0,01. S touto hodnotou byla přesnost modelu 87,55 %, dosažené zlepšení je 0,39 %. Hodnoty od 0,065 až do

0,12 měly stejnou nebo velmi podobnou přesnost modelu jako základní hodnota parametru (0,1). Při zvyšování parametru nad 0,12 přesnost modelu prudce klesala.



Obrázek 22: Přesnost modelu při různých hodnotách parametru „convergence rate“ u metody SDCA na datové sadě IMDB

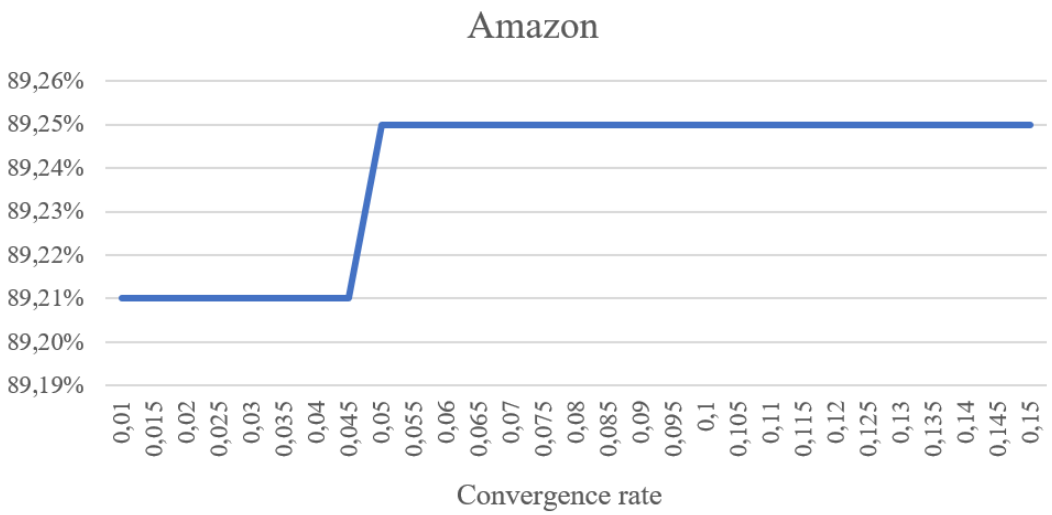
U datové sady Amazon k žádnému zlepšení nedošlo. Mezi hodnotami 0,15 a 0,05 byla přesnost modelu konstantní. Při dalším snižovaní parametru *convergence rate* přesnost klesnula o 0,04 % a byla opět stejná pro všechny další hodnoty. Metoda SDCA pravděpodobně konverguje vždy do stejné hodnoty, změnou parametru se pouze reguluje rychlost konvergence a při nízkých hodnotách se metoda ukončí před úplnou konvergencí nebo uvízne v lokálním minimu loss funkce. Kompletní výsledky testu lze pozorovat na obrázku 23.

6.3.2 Averaged Perceptron

Pro metodu Averaged Perceptron jsou testovány 3 různé cost funkce a u každé z nich různé parametry *learning rate* a počet iterací. Learning rate zpomaluje učení modelu. Než se upraví váhy perceptronu tak se hodnota, o kterou se mají upravit, vynásobí parametrem *learning rate*. Tento parametr může nabývat hodnot v intervalu $(0;\ 1\rangle$, základní hodnota je 1 a k tomu jsou testovány ještě následující hodnoty $\{0,05; 0,1; 0,2; 0,5\}$. Postupné snižování tohoto parametru je zakázáno. Počet iterací je v základu nastaven na 1, další testované hodnoty jsou 10, 50, 100, 200 a 400. První cost funkce Hinge loss je definována rovnicí (16), kde $\hat{y}$ je predikované skóre a y je správná predikovaná třída (-1, 1).

$$L(\hat{y},\ \mathrm{y}) = max(0, 1 - y\hat{y}) \tag{16}$$

$$L(p(\hat{y}), y) = -yln(\hat{y}) - (1 - y)lnt(1 - \hat{y}) \tag{17}$$



Obrázek 23: Přesnost modelu při různých hodnotách parametru „convergence rate“ u metody SDCA na datové sadě Amazon

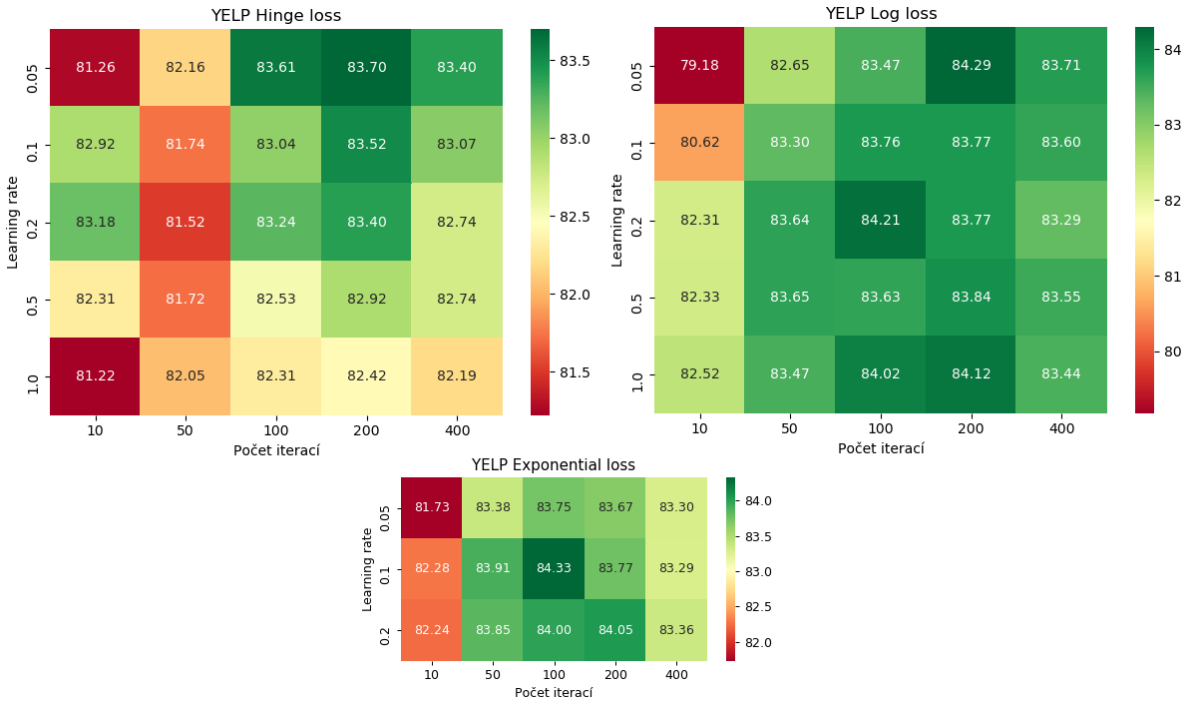
Další cost funkce je Log loss, definice této funkce je rovnice viz (17). Proměnná $\hat{y}$ je predikované skóre, $p(\hat{y})$ je pravděpodobnost, že predikovaná třída bude pozitivní a y je správná predikovaná třída (0, 1). Funkce je konvexní a roste lineárně pro negativní hodnoty, tudíž je méně citlivá pro odlehlé hodnoty (angl. *outliers*). Poslední cost funkce bude Exponential loss, jejíž definice je:

$$L(\hat{y}, \mathrm{y}) = e^{-\beta y \hat{y}} \tag{18}$$

Správná predikovaná třída y zde nabývá hodnot -1 a 1. β je konstanta sloužící k redukci vysokých hodnot, která je v základu nastavena na 1. Tato cost funkce je konvexní a narůstá exponenciálně pro negativní hodnoty, což ji dělá více citlivou pro odlehlé hodnoty. Tuto cost funkci využívá například známy algoritmus AdaBoost.

Jak můžeme vidět na obrázku 24, u první datové sady Yelp s funkcí Hinge loss se zvyšováním počtu iterací výsledky zlepšují až do 200 iterací, při 400 iteracích je už model pravděpodobně přeučený a přesnost modelu se snižuje. Přeučení nastává, když se model naučí příliš dobře predikovat trénovací data a přijde o schopnost generalizace neboli schopnost klasifikovat data, která ještě neviděl. Od 50 a více iterací je dosaženo nejlepších výsledků s hodnotou learning rate 0,05. S nárůstem počtu iterací narůstá také složitost algoritmu, a tedy délka trvání učení modelu, proto můžeme považovat přesnost modelu 83,18 %, která byla naměřena pouze při 10 iteracích s parametrem learning rate 0,2 za dostatečnou. Nejlepší výsledek 83,70 %, s cost funkcí Hinge loss je dosažen při 200 iteracích a s parametrem learning rate 0,05.

Cost funkce Log loss má velmi dobré výsledky i při menším počtu iterací. Větší počet iterací téměř vždy zvýší přesnost modelu s výjimkou 400 iterací, kde je model už přeučený. Nejlepší výsledek byl naměřen se stejnými parametry jako v případě předchozí přechodové funkce a to



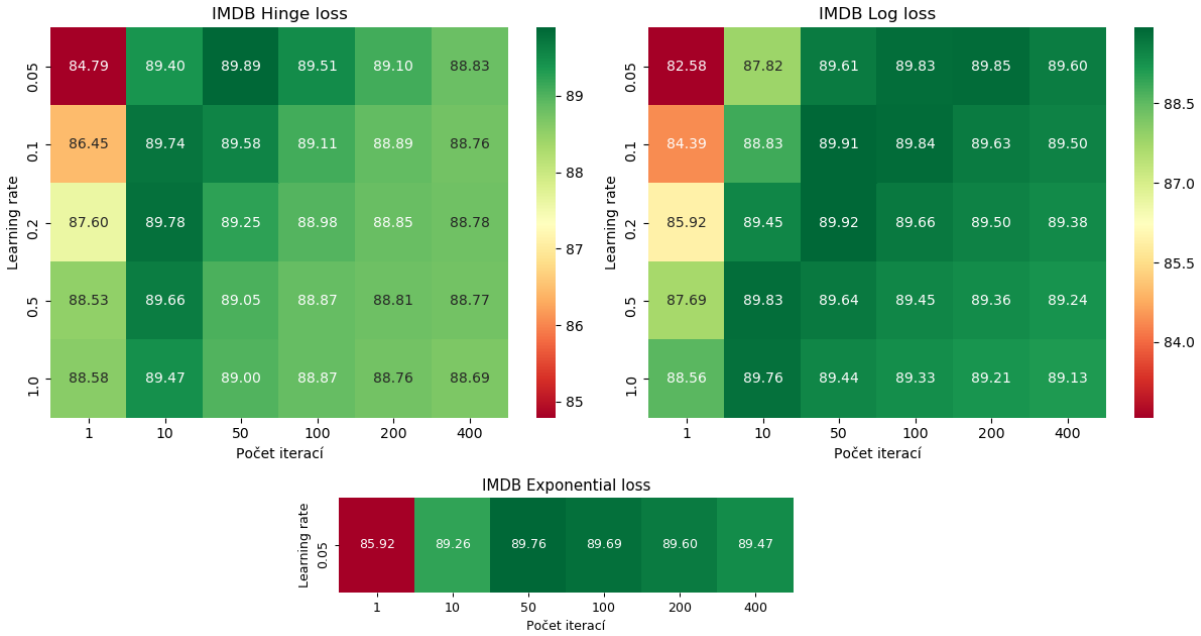
Obrázek 24: Přesnost modelu pro různé cost funkce s datovou sadu Yelp (v procentech)

84,29 %. Při hodnotě learning rate 0,2 byl změřen velmi podobný výsledek 84,21 %, ale s polovičním počtem iterací.

Jako nejlepší cost funkce se ukázala Exponential loss, která pouze se 100 iteracemi a hodnotou learning rate 0,1 dosahuje přesnosti modelu 84,33 %, čímž byl překonán nejlepší výsledek metody SDCA. Tato exponenciální funkce nám dává velmi vysoké hodnoty pro odlehlá pozorování, kvůli tomu se hodnoty vah rychle zvyšují a nebylo možné změřit přesnost pro hodnoty parametru learning rate 0,5 a 1.

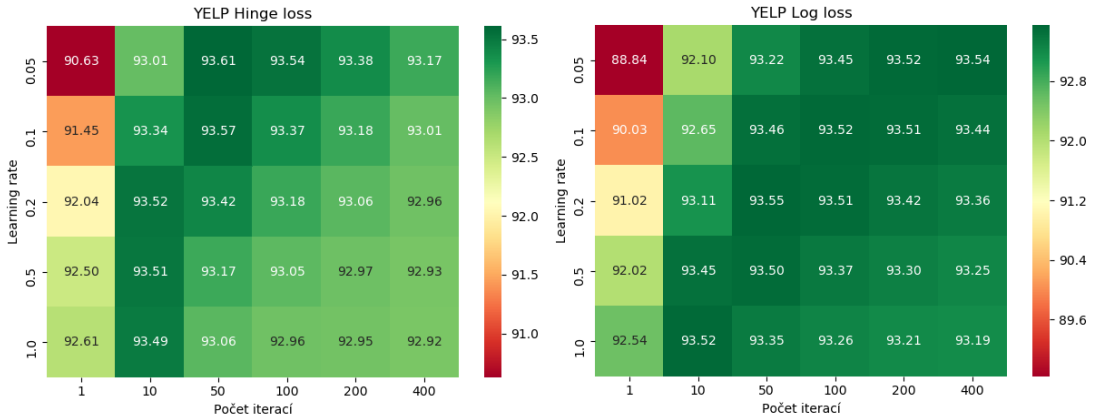
Výsledky předchozí datové sady jsou spíše neuspořádané a nemůžeme v nich vidět jednoznačné závislosti mezi hyperparametry. Na výsledcích z větší datové sady IMDB je možné pozorovat, jak parametr *learning rate* ovlivňuje počet iterací, které potřebujeme, aby model dosáhnul co nejlepších výsledků a taky kolik může proběhnout iterací, než se model přeučí. Nejlepšího výsledku 89,92 % bylo dosaženo s cost funkcí Log loss při 50 iteracích a s parametrem *learning rate* 0,2. Výsledky experimentu můžeme vidět na obrázku 25. S použitím cost funkce Hinge loss byly výsledky podobné, akorát byl model více citlivý na přeučení a s vyšším počtem iterací byl pokles přesnosti znatelnější. Průběh algoritmu při využití cost funkce Exponential loss skončil chybou pro všechny hodnoty parametru *learning rate* s výjimkou hodnoty 0,05. Důvod této chyby byl stejný jako u předchozí datové sady. V jediné naměřené hodnotě parametru *learning rate* byly výsledky horší než u zbylých dvou cost funkcí.

U datové sady Amazon byl průběh testování podobný jako v případě IMDB s rozdílem, že přeučení modelu přicházelo až při více iteracích (viz graf 9). S cost funkcí Hinge loss k přeučení



Obrázek 25: Přesnost modelu pro různé cost funkce s datovou sadu IMDB (v procentech)

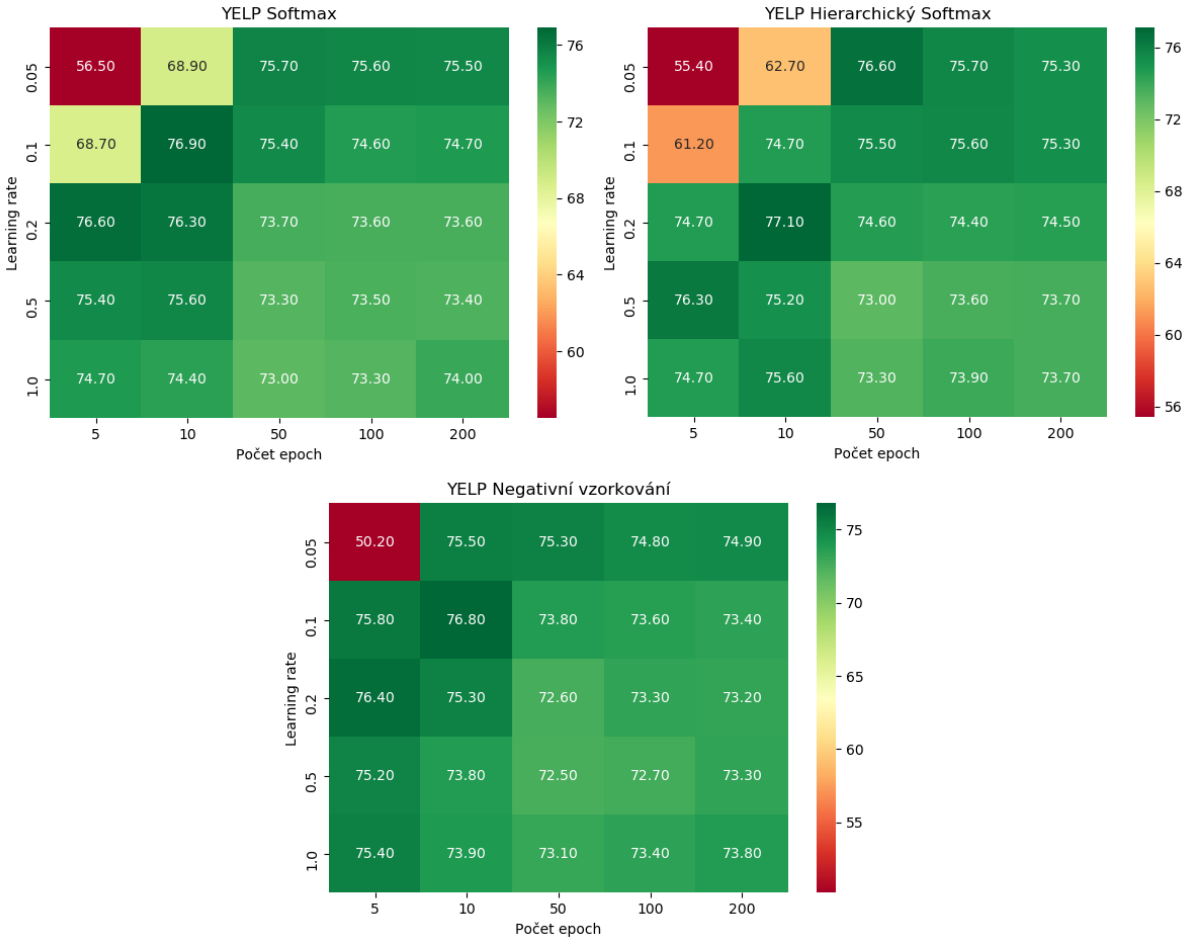
docházelo už při 50 iteracích v kombinaci s hodnotami *learning rate* 1,0 až 0,2 a při 100 iteracích u nižších hodnot. Nejlepší výsledek 93,61 % byl naměřen při 50 iteracích s *learning rate* 0,05. Na výsledcích druhé cost funkce Log loss lze pozorovat zpomalování učení modelu se snižujícím se parametrem *learning rate*. V případě nejnižší hodnoty 0,05 nedošlo k přeučení ani při 400 iteracích a je možné, že by se přesnost dále zlepšovala, ale s takto velkou datovou sadou je další zvyšování iterací výpočetně velice náročné. Nejlepší výsledky se u všech hodnot *learning rate* jen nepatrně lišily, a zatímco u hodnoty 1 byla dosažena nejlepší přesnost už u 10 iterace, tak u hodnoty 0,05 byla stejná přesnost dosažena až při 200 iteracích. To ukazuje výhodu větší hodnoty *learning rate*, jelikož můžeme dosáhnout stejnou přesnost s 20krát menší výpočetní náročností.



Obrázek 26: Přesnost modelu pro různé cost funkce s datovou sadu Amazon (v procentech)

6.3.3 FastText

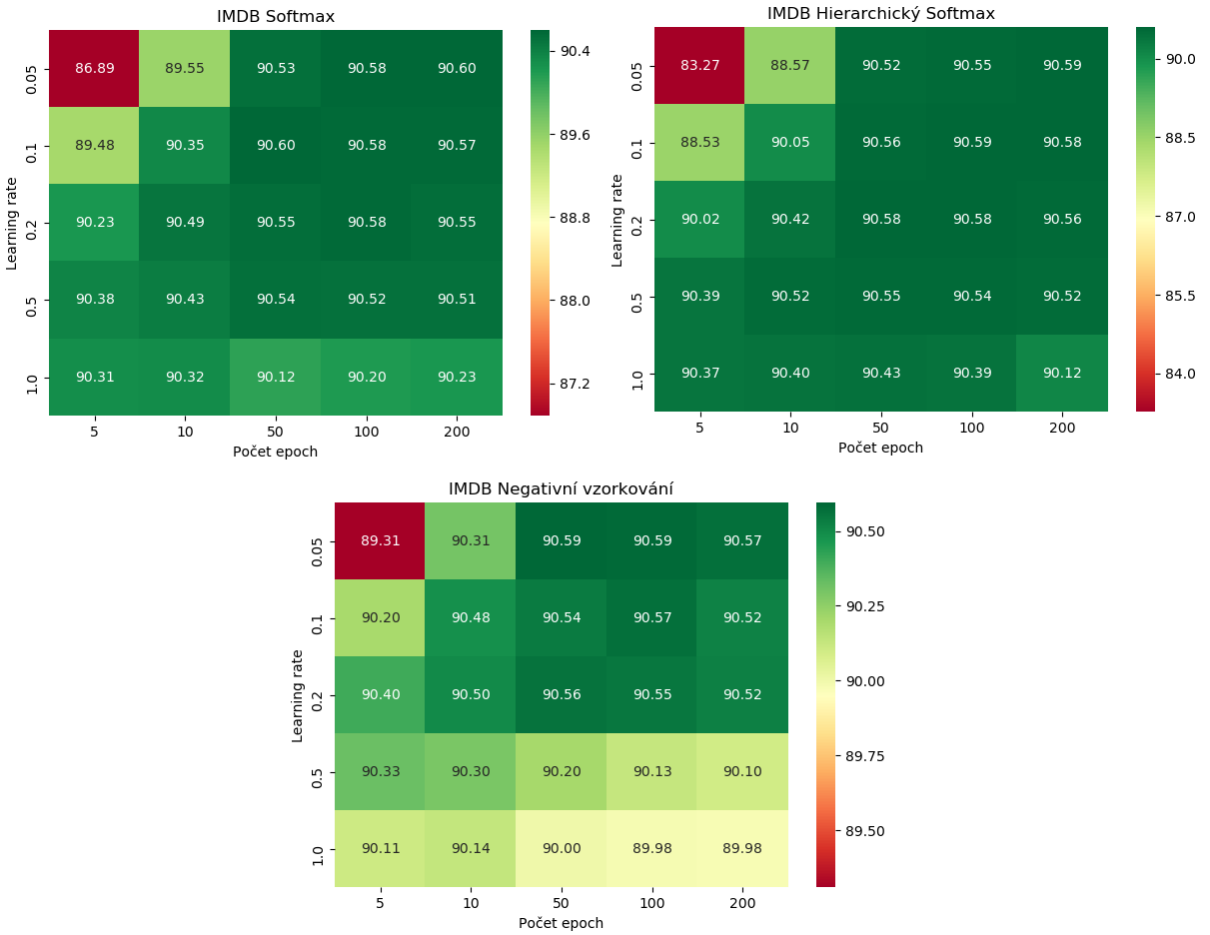
Stejně jako předchozí metoda i FastText spadá do kategorie neuronových sítí, a proto byly opět testovány parametry *learning rate*, počet epoch a rozdílné cost funkce. Learning rate byl testován v hodnotách {0,05; 0,1; 0,2; 0,5;1}, přičemž základní hodnota je 0,1. S ohledem na časovou náročnost trénování bylo testování prováděno při počtu epoch 5, 10, 50, 100 a 200. Výchozí hodnota byla 5 epoch. Dále byl testován vliv tří cost funkcí, Softmax, Hierarchický softmax a Negativní vzorkování (jejich popis se nachází v kapitole 4.3). V základu se používá funkce Softmax, další 2 funkce jsou určeny pro urychlení trénování modelu při zanedbatelném snížení přesnosti. Vzhledem k tomu mělo význam měřit také čas trénování a následně tyto časy porovnat. Počet negativních vzorků byl pro obě datové sady nastaven na hodnotu 5.



Obrázek 27: Přesnost modelu pro různé cost funkce s metodou FastText a datovou sadou Yelp (v procentech)

Jako první byl proveden tento experiment s datovou sadou Yelp (viz obrázek 27). Výsledky se všemi cost funkcemi byly podobné, nejlépe však dopadla překvapivě metoda Hierarchický softmax. Tato funkce je pouze aproximací funkce Softmax, tudíž by měla snížit přesnost mo-

delu. Tento nečekaný výsledek byl způsoben tím, že je datová sada velmi malá a trénování modelu je velmi citlivé pro jakékoliv změny. Všechny tři cost funkce dosahovaly největší efektivity v rozsahu 5–50 epoch, při více epochách zůstávala přesnost stejná nebo se snižovala. Nejlepší výsledek 77,10 % byl dosažen s parametrem *learning rate* 0,2 při 10 epochách se zmíněnou funkcí Hierarchický softmax. Přestože bylo dosaženo výrazného zlepšení, tak oproti ostatním metodám je přesnost modelu stále velmi nízká. Doba trénování modelu u této datové sady byla pro všechny tři funkce velmi nízká, pouze s nepatrnými rozdíly mezi funkcemi, a proto nemá význam tyto hodnoty porovnávat.



Obrázek 28: Přesnost modelu pro různé cost funkce s metodou FastText a datovou sadou IMDB (v procentech)

Na obrázku 28 můžeme vidět výsledky pro datovou sadu IMDB. Nejlépe si vedla základní funkce Softmax. Při hodnotě parametru *learning rate* 0,1 a 50 epochách bylo dosaženo přesnosti modelu 90,60 %. Model nejevil žádné známky přeučení ani při vysokých hodnotách epoch, ale po 50 epochách byly výsledky stejné nebo horší. Toho je docíleno tím, že parametr *learning rate* se postupně snižuje, dokud neklesne až na nulu. Poté se učení modelu zastaví bez ohledu na nastavený počet epoch. Snižování tohoto parametru je ale ovlivněno nastaveným počtem epoch,

proto se výsledky mírně liší a v případě vyššího parametru *learning rate* můžeme vidět výraznější pokles přesnosti při 200 epochách.

S funkcí Hierarchický softmax (dále jen HS) bylo dosaženo podobných výsledků jako s předešlou funkcí, avšak průměrná doba trénování modelu byla významně kratší (viz tabulka 17). Zatímco s funkcí Softmax trvalo trénovaní modelu při 200 epochách v průměru 82,2 minut, použitím HS se tato doba zkrátila na 71,2 minut. Funkce Negativní vzorkování (dále jen NV) také dosáhla podobných výsledků se zanedbatelným rozdílem, ale úspora času byla jen mírná. Průměrná doba trénování modelu se zkrátila na 80,3 minut. Ačkoliv je datová sada vybalancovaná a funkce NV je určená pro takové datové sady, tak kvůli křížové validaci není zaručeno, že v trénovacích datech budou obě klasifikované třídy ve stejném poměru. Díky tomu je naopak zvýhodněna funkce HS, přesto jsou obě funkce určeny pro klasifikaci do většího množství tříd, kde mohou dosáhnout daleko větší úspory času.

Funkce	Softmax	HS	NV
Průměrná doba trénování	82,2	**71,2**	80,3

Tabulka 17: Průměrná doba trénování modelu při 200 epochách s datovou sadou IMDB (v minutách)

Experiment s datovou sadou Amazon ukázal, jak rychle se model dokáže učit. Model dosahuje nejlepších výsledků při pouhých 5 epochách, s dalším trénováním začíná být přeučený a přesnost se snižuje u všech testovaných hodnot learning rate. Kompletní výsledky jsou na obrázku 29, lze na nich pozorovat, jak se zvyšováním parametru learning rate a počtu epoch zhoršuje kvalita modelu. Nejvyšší přesnost 93,38 % byla dosažena s learning rate 0,05 a cost funkcí Softmax při 5 epochách. Rozdíly v přesnosti s ostatními cost funkcemi byly zanedbatelné. Funkce HS měla horší přesnost pouze o 0,03 % a negativní vzorkovaní o 0,12 %.

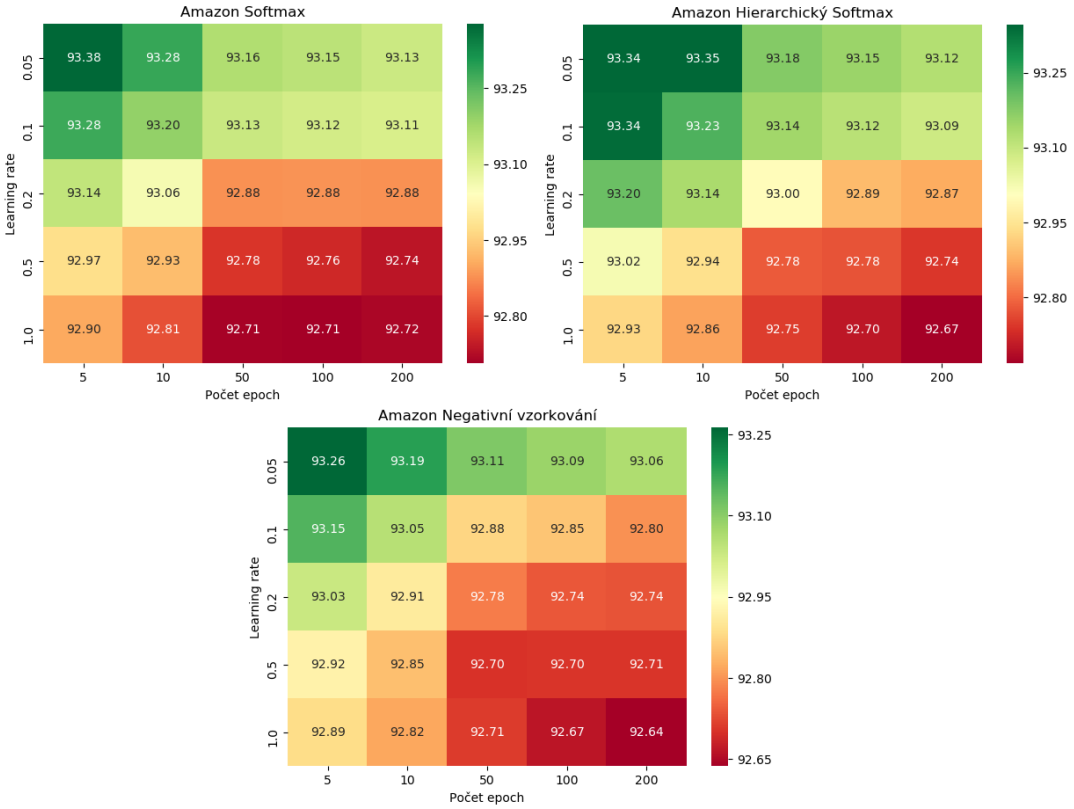
Z hlediska časové náročnosti byla funkce HS při 200 epochách přibližně o 15 % rychlejší (viz tabulka 18). Funkce negativního vzorkování byla o něco pomalejší i přesto, že je datová sada vybalancovaná a nebyla provedena křížová validace. Nejlepších výsledku ale bylo dosáhnuto při 5 epochách a zde je časová úspora aproximačních funkcí bezvýznamná.

Funkce	Softmax	HS	NV
Průměrná doba trénování	299	**254**	268

Tabulka 18: Průměrná doba trénování modelu při 200 epochách s datovou sadou Amazon (v minutách)

6.3.4 Neuronové sítě

Tento experiment zkoumá, jaký vliv má změna architektury dané sítě na výsledný model. Nové architektury mají jednak více jednotek (neuronů) v aktuálních vrstvách, a taktéž jsou přidány



Obrázek 29: Přesnost modelu pro různé cost funkce s metodou FastText a datovou sadou Amazon (v procentech)

nové vrstvy. Nové vrstvy jsou určeny především k normalizaci a regularizaci dat při trénování sítě, jež slouží ke zpomalení nežádoucího jevu přeučení sítě. V nových architekturách jsou použity tyto vrstvy:

- **Dropout** – Tato jednoduchá regularizační technika je aplikována pouze při procesu učení neuronové sítě. Cílem techniky je snížení závislosti na specifických váhách neuronů. Funguje na principu náhodné deaktivace neuronů v síti (nebo v určených vrstvách) pro každou trénovací instanci. Díky tomu se do trénovacích dat vloží šum a zvýší se míra abstrakce neuronové sítě. Přidáním dropout vrstvy je tato technika aplikována na předchozí vrstvu. Tato vrstva umožňuje nastavit parametrem pravděpodobnost deaktivace neuronu. Po ukončení trénování je tato vrstva zrušena a během evaluace nedochází k deaktivaci neuronů.
- **Batch normalization** – Poměrně nová technika (2015) pro zlepšení výkonu a stability neuronových sítí. Princip této techniky spočívá v normalizaci výstupů jednotlivých vrstev sítě pomocí centralizace hodnot okolo nuly a změnou měřítka. Hodnoty pro normalizaci se počítají pouze z aktuální dávky dat. Batch normalization vrstva přidává do sítě také trochu šumu a způsobuje tím mírnou regularizaci.

LSTM

První testovaná LSTM architektura pro tento experiment se skládá z desíti vrstev. Oproti původní struktuře jsou tvořeny word embeddings s 64 parametry (původní hodnota 16) a LSTM vrstva se skládá ze 100 jednotek (původní hodnota 16). Navíc jsou přidány další dvě Batch normalization vrstvy a dvě Dropout vrstvy. Kompletní architektura je popsána v tabulce 19, konstanta m je délka všech recenzí po zarovnání (angl. *padding*) a hodnota v závorce u Dropout vrstev je pravděpodobnost deaktivace neuronu.

Název vrstvy	Tvar výstupu	Aktivační funkce
Vstupní vrstva	m	
Embedding	$m \times 64$	
Batch normalization	$m \times 64$	
Dropout (0,4)	$m \times 64$	
Bidirectional LSTM	$m \times 200$	Tanh
Global max-pooling 1D	*200*	
Batch normalization	*200*	
Dense	*100*	lineární f.
Dropout (0,3)	*100*	
Dense	*1*	Sigmoid

Tabulka 19: Architektura první rozšířené LSTM sítě

Ve druhé navržené architektuře je cíleno na kombinaci dvou obousměrných LSTM vrstev. Síť je složena ze 14 vrstev, jež navazují postupně za sebou tak, jak je uvedeno v tabulce 20. Vektorová reprezentace termů má opět podobu vektorů s 64 dimenzemi. Obě LSTM vrstvy mají 64 skrytých jednotek a aktivační funkci Tanh. Mezi těmito vrstvami je navíc Batch normalization vrstva, Dropout vrstva s pravděpodobností deaktivace 30 % a Max-pooling vrstva, která slučuje dvě hodnoty do jedné.

Výsledky experimentu obou rozšířených LSTM architektur se nachází v tabulce 21. U datové sady Yelp jsou výsledky obou sítí horší než u původní sítě. První síť snížila přesnost modelu o 1 % a druhá síť, jež je hlubší, snížila přesnost ještě více. To dokazuje, že hlubší síť nemusí vždy zvýšit přesnost modelu, naopak jí může zhoršit. S datovou sadou IMDB mělo rozšíření sítě pozitivní vliv na přesnost klasifikace. Druhá architektura se dvěma LSTM vrstvami měla vyšší přesnost o téměř jedno procento, konkrétně 90,68 %. První architektura si vedla ještě lépe s přesností 91,13 %, tímto výsledkem byly překonány doposud všechny ostatní metody. V případě datové sady Amazon bylo dosaženo nejlepšího výsledku s první sítí, jejíž přesnost byla 95 %. S druhou sítí nebylo možné provést experiment, jelikož učení sítě vždy skončilo neznámou chybou frameworku Tensorflow.

Název vrstvy	Tvar výstupu	Aktivační funkce
Vstupní vrstva	m	
Embedding	$m \times 64$	
Batch normalization	$m \times 64$	
Dropout (0,4)	$m \times 64$	
Bidirectional LSTM	$m \times 128$	Tanh
Batch normalization	$m \times 128$	
Max-pooling 1D (2)	$(m/2) \times 128$	
Dropout (0,3)	$(m/2) \times 128$	
Bidirectional LSTM	$(m/2) \times 128$	Tanh
Global max-pooling 1D	*128*	
Batch normalization	*128*	
Dense	*100*	lineární f.
Dropout (0,3)	*100*	
Dense	*1*	Sigmoid

Tabulka 20: Architektura druhé rozšířené LSTM sítě

	LSTM 1	LSTM 2
Yelp Dataset	80,50 %	79,70 %
IMDB Dataset	91,13 %	90,68 %
Amazon Dataset	95,00 %	N/A

Tabulka 21: Přesnost modelů rozšířených LSTM architektur na všech datových sadách

CNN

V kategorii konvolučních sítí jsou testovány opět dvě rozšířené architektury. První z nich má menší počet vrstev skládajících se z mnoha jednotek. Je složena z osmi vrstev, které jsou poskládány za sebou. Celá struktura sítě je v tabulce 22, hodnota v závorce za konvoluční vrstvou je velikost konvolučních filtrů. Embedding vrstva tvoří word embeddings s 50 parametry (původní hodnota 16). Poté následuje Dropout vrstva s parametrem pro deaktivaci neuronů 0,3. Konvoluční vrstva se skládá z 250 filtrů o velikost 3, zpracovává tedy 3 termy najednou. Pro aktivaci je použita funkce Relu. Následuje Global max-pooling vrstva, jež sníží dimenzi výstupu. Na konci je jedna Dense vrstva s 250 neurony a druhá pro formulaci výstupu sítě s jedním neuronem, jehož aktivační funkcí je Sigmoid. Mezi nimi je Dropout vrstva s nižší pravděpodobností deaktivace neuronu 20 %.

Druhá síť je naopak orientovaná spíše do hloubky (viz tabulka 23). Obsahuje 18 menších vrstev, které se dají rozdělit do tří častí. Vstupní část, konvoluční bloky a výstupní část. Za

Název vrstvy	Tvar výstupu	Aktivační funkce
Vstupní vrstva	m	
Embedding	$m \times 50$	
Dropout (0,3)	$m \times 50$	
Konvoluční 1D (3)	$(m-2) \times 250$	Relu
Global max-pooling 1D	250	
Dense	250	Relu
Dropout (0,2)	250	
Dense	1	Sigmoid

Tabulka 22: Architektura první rozšířené konvoluční sítě

vstupní vrstvou je Embedding vrstva, jejíž výstup je matice $m \times 64$, kde m je počet slov recenze a hodnota 64 určuje počet popisujících parametrů pro každý term. Následuje Batch normalization vrstva a Dropout vrstva s parametrem 0,3. Dále jsou čtyři konvoluční bloky, které se skládají z konvoluční vrstvy, Batch normalization vrstvy a Dropout vrstvy s výjimkou posledního bloku, který neobsahuje Batch normalization vrstvu. Konvoluční filtry v prvním bloku pokrývají 7 termů najednou, v ostatních blocích mají filtry velikost pouze 3. Všechny bloky mají jako aktivační funkci Relu a Dropout vrstvy mají pravděpodobnost deaktivace jednotky (filtru) 30 %. Vstupní matice jsou před konvolucí rozšířeny o nulové řádky tak, aby matice měla po konvoluci stejnou velikost (tzv. *padding*). Po těchto čtyřech blocích je ještě jedna konvoluční vrstva se dvěma filtry o velikosti 2 a lineární aktivační funkcí. Tu následuje Global average-pooling vrstva, jež sníží dimenzi výstupu tak, že v každém sloupci matice vypočte průměrnou hodnotu a z těchto hodnot poskládá výsledný vektor. Na konci sítě je Dense vrstva s jedním neuronem a aktivační funkcí Sigmoid.

V tabulce 24 jsou výsledky přesnosti klasifikace obou CNN architektur. U datové sady Yelp měla první rozšířená konvoluční síť o půl procenta lepší výsledek než ta základní, v provedeném testu dosáhla přesnosti 82,90 %. Druhá architektura naopak přesnost klasifikace snížila. Pro datovou sadu IMDB znamenaly obě rozšířené sítě zlepšení. Lépe ale dopadla druhá konvoluční síť s přesností klasifikace 90,77 %, zlepšení bylo téměř o jedno procento. Testy s datovou sadou Amazon ukázaly významné zlepšení u obou architektur. S první sítí byla dosažena přesnost klasifikace 93,95 % a s druhou sítí byla přesnost 94,44 %, tzn. zlepšení o 1,37 % oproti základní CNN.

Název vrstvy	Tvar výstupu	Aktivační funkce
Vstupní vrstva	m	
Embedding	$m \times 64$	
Batch normalization	$m \times 64$	
Dropout (0,3)	$m \times 64$	
Konvoluční 1D (7)	$m \times 32$	Relu
Batch normalization	$m \times 32$	
Dropout (0,3)	$m \times 32$	
Konvoluční 1D (3)	$m \times 32$	Relu
Batch normalization	$m \times 32$	
Dropout (0,3)	$m \times 32$	
Konvoluční 1D (3)	$m \times 32$	Relu
Batch normalization	$m \times 32$	
Dropout (0,3)	$m \times 32$	
Konvoluční 1D (3)	$m \times 32$	Relu
Dropout (0,3)	$m \times 32$	
Konvoluční 1D (2)	$(m-1) \times 2$	lineární f.
Global average-pooling 1D	2	
Dense	1	Sigmoid

Tabulka 23: Architektura druhé rozšířené konvoluční sítě

	CNN 1	CNN 2
Yelp Dataset	82,90 %	80,20 %
IMDB Dataset	90,50 %	90,77 %
Amazon Dataset	93,95 %	94,44 %

Tabulka 24: Přesnost modelů rozšířených CNN architektur na všech datových sadách

6.4 Shrnutí výsledků

V této sekci je uvedeno shrnutí nejlepších výsledků všech metod pro analýzu sentimentu a jejich porovnání na všech datových sadách. Nejlepší dosažené výsledky jsou vypsány v tabulce 25.

	Yelp Dataset	IMDB Dataset	Amazon Dataset
Averaged Perceptron	**84,33 %**	89,92 %	93,61 %
SDCA	83,57 %	87,55 %	89,25 %
FastText	77,10 %	90,60 %	93,38 %
LSTM	81,50 %	**91,13 %**	**95,00 %**
CNN	82,90 %	90,77 %	94,44 %

Tabulka 25: Nejlepší dosažená přesnost modelů jednotlivých algoritmů na všech datových sadách (v procentech)

U nejmenší datové sady Yelp bylo v prvním experimentu (metody v základním nastavení) dosaženo nejlepšího výsledku 82,24 % s metodou SDCA. Po úpravě techniky pro vektorizaci textu a následné optimalizaci parametrů se přesnost modelu zvýšila na 83,57 %. Tento výsledek byl překonán metodou Averaged Perceptron s finální přesností klasifikace 84,33 %. Pro dosažení tohoto výsledku byla v rámci předzpracování datové sady odstraněna interpunkce z textu a změněna metodika pro hodnocení relevance slov v technice Bag of words. Při optimalizaci hyperparametrů byla změněna loss funkce, zvýšen počet epoch a snížen parametr learning rate. Neuronové sítě překonaly hranici přesnosti 80 %, jediná výjimka byla metoda FastText, jejíž nejlepší dosažený výsledek byl o více než 7 % horší.

Na datové sadě IMDB s nejdelšími dokumenty (recenzemi) dominovaly neuronové sítě. V základním nastavení měla nejlepší výsledek konvoluční síť s přesností klasifikace 89,72 %. Obě sítě se zlepšily po lemmatizaci datové sady. Následně byla konvoluční síť překonána obousměrnou rekurentní sítí s jednotkami LSTM díky rozšíření architektury sítě. Nejlepší výsledek 91,13 % byl dosažen po zvětšení rekurentní vrstvy a přidání regularizačních vrstev. Nejhůře dopadla metoda SDCA s nejvyšší dosaženou přesností modelu 87,55 %.

V případě největší datové sady Amazon byla nejlepší metoda rekurentní síť LSTM. Tato neuronová síť dosáhla nejlepších výsledků i v původní konfiguraci s přesností 94,12 %. Žádná technika předzpracování datové sady nezvýšila její přesnost klasifikace. Nejvyšší přesnost 95,00 % byla dosažena, stejně jako v předchozím případě, s první rozšířenou LSTM sítí. Nejnižší přesnost modelu měla opět metoda SDCA, se kterou bylo dosaženo nejlepšího výsledku 89,25 %.

7 Závěr

Cílem této práce bylo vyhledat a otestovat metody pro analýzu sentimentu na vhodných datových sadách. Pro splnění tohoto úkolu bylo vybráno pět metod pro analýzu dat z oblasti strojového učení na základě aktuálních trendů. První dvě metody se řadí do kategorie lineárních klasifikátorů a zbylé tři jsou z kategorie neuronových sítí, jejich principy byly detailně popsány ve čtvrté kapitole. Následně byly vybrány tři datové sady, z nichž jedna reprezentuje velmi malou datovou sadu, druhá datová sada má běžnou velikost s velmi dlouhými instancemi a poslední datová sada je mnohonásobně větší než předešlé. Jako první experiment byly otestovány všechny metody v základním nastavení s pouze nezbytným předzpracováním textu. Už v základním nastavení měly některé metody dostačující výsledky a byly použitelné bez dalších úprav.

V dalším experimentu bylo otestováno, jaký má vliv pokročilejší předzpracování textu na přesnost klasifikace. Na nejmenší datové sadě nebylo předzpracování textu příliš efektivní. Ačkoliv některé metody dosáhly zlepšení, tak přesnost jejich klasifikátoru byla stále nízká. Výjimkou byla konvoluční síť, jež překonala nejlepší výsledek díky lemmatizaci datové sady. U střední datové sady už mělo předzpracování textu většinou pozitivní dopad. Pouze u neuronových sítí měly všechny techniky kromě lemmatizace za následek snížení přesnosti. Překvapivý výsledek experimentu byl s datovou sadou Amazon, kde byl vliv předzpracování textu většinou negativní. V některých případech nastalo mírné zlepšení přesnosti, ale i tak byly výsledky horší oproti ostatním metodám.

Třetí experiment popisuje změny technik při transformaci textu do vektorové reprezentace. V případě metody Bag of Words se testovaly různé metodiky pro ohodnocení termů a rozdělení textu na n-gramy různých velikostí. U nerunových sítí byly spojovány slova do frází různých velikostí, tato technika bohužel nepřinesla zlepšení u žádné datové sady. Zatímco změna metodiky ohodnocení termů měla ve všech případech pozitivní vliv, tak použitím n-gramů se přesnost zvýšila až u největší datové sady. Výjimkou byla metoda FastText, která už u prostřední datové sady prokázala zlepšení při použití bigramů.

Následující experiment se zabývá optimalizací parametrů testovaných metod. Optimalizace byla provedena technikou grid search, jež testuje všechny kombinace parametrů z předdefinovaných množin. U metody SDCA byl optimalizován pouze jeden parametr, který slouží pro regulaci rychlosti konvergence. Klasifikátory metod Averaged Perceptron a FastText fungují na podobném principu, proto u nich byly optimalizovány stejné parametry, konkrétně počet iterací, loss funkce a parametr regulující rychlost učení. Téměř ve všech případech byly nalezeny lepší parametry než v původním nastavení, nejmenší zlepšení bylo u metody SDCA. V případě neuronových sítí byly optimalizovány jejich architektury. Změny architektur se týkaly zvětšení původních vrstev, přidání regularizačních vrstev a konvolučních nebo LSTM vrstev. Rozšířené architektury dosáhly lepších výsledků především u dvou větších datových sad. U nejmenší datové sady přinesla zlepšení pouze jedna rozšířená architektura konvoluční sítě.

Z finálních výsledků vyplývá, že velikost trénovacích dat je nejdůležitější faktor všech metod strojového učení. Předzpracováním textu nebo optimalizací lze dosáhnout lepších výsledků, tyto techniky ovšem nemohou výrazně vylepšit nekvalitní datovou sadu. Problém s nedostatkem trénovacích dat by se dal částečně řešit použitím předtrénovaného modelu. Metoda SDCA se hodí spíše pro menší datové sady a použitá implantace dosahovala dobrých výsledků i bez změny hyperparametrů nebo bez pokročilých technik pro předzpracování textů. Lineární klasifikátor Averaged Perceptron je velmi jednoduchá technika a dosáhla nadprůměrných výsledků na všech datových sadách v těsném závěsu ostatních metod. Metoda FastText je založena na tvorbě word embeddings, jež představují moderní přístup pro vektorovou reprezentaci textu. Jak ukazují výsledky, tato metoda není příliš vhodná pro malé datové sady, na nichž je problém vytvořit kvalitní vektory charakterizující jednotlivá slova. Neuronové sítě jsou naopak vhodné pro všechny typy dat, ale u menších datových sad mohou mít zbytečně moc hluboké sítě negativní vliv na přesnost modelu. Experimenty také ukázaly, že předzpracování textu pro neuronové sítě má většinou neblahý efekt nebo způsobí jen zanedbatelné zlepšení.

Obě neuronové sítě dosahovaly velmi dobrých výsledků, i když měly jednoduché architektury a nebylo potřeba žádné techniky pro předzpracování dat, pouze ty nezbytně nutné pro fungování metod. Proto jsou v dnešní době umělé neuronové sítě tak populární. Také bylo ověřeno, že konvoluční sítě nejsou určeny pouze pro zpracování obrazu, ale dosahují výborných výsledků i v oblasti zpracování textů.

Pro další zvýšení přesnosti klasifikátorů, pomocí předzpracování textu, bychom mohli upravit seznamy stop slov pro konkrétní datovou sadu a vyzkoušet jiné nástroje pro lemmatizaci. Další možností by bylo použít již vytvořenou vektorovou reprezentaci slov v podobě word embeddings pro vybraný jazyk. Tyto předpřipravené modely jsou volně dostupné z platforem jako Word2Vec, FastText, GloVe atd. Tato reprezentace by poté sloužila jako vstup pro všechny testované metody. Mimo předzpracování textu by se dalo docílit zlepšení klasifikace pomocí hlubokých předtrénovaných modelů, které se pouze doučí na konkrétní řešenou úlohu.

Literatura

1. PANG, Bo; LEE, Lillian. Opinion Mining and Sentiment Analysis. *Foundations and Trends® in Information Retrieval*. 2008, roč. 2, č. 1–2, s. 1–135. ISSN 1554-0669. Dostupné z DOI: 10.1561/1500000011.
2. PETKEVIČ, Vladimír. KOMPUTAČNÍ LINGVISTIKA. *In: Petr Karlík, Marek Nekula, Jana Pleskalová (eds.), CzechEncy – Nový encyklopedický slovník češtiny*. 2017.
3. NASUKAWA, Tetsuya; YI, Jeonghee. Sentiment analysis: Capturing favorability using natural language processing. In: 2003-01, s. 70–77. Dostupné z DOI: 10.1145/945645.945658.
4. VESELOVSKÁ, Kateřina. POSTOJOVÁ ANALÝZA. *In: Petr Karlík, Marek Nekula, Jana Pleskalová (eds.), CzechEncy – Nový encyklopedický slovník češtiny*. 2017.
5. JEYAPRIYA, A.; SELVI, C. S. K. Extracting aspects and mining opinions in product reviews using supervised learning algorithm. In: *2015 2nd International Conference on Electronics and Communication Systems (ICECS)*. 2015, s. 548–552.
6. RASCHKA, Sebastian. *Naive Bayes and Text Classification I - Introduction and Theory*. 2014. Dostupné z arXiv: 1410.5329 [cs.LG].
7. JEYAPRIYA, A.; SELVI, C. S. K. Extracting aspects and mining opinions in product reviews using supervised learning algorithm. In: *2015 2nd International Conference on Electronics and Communication Systems (ICECS)*. 2015, s. 548–552.
8. KANARIS, Ioannis; KANARIS, Konstantinos; HOUVARDAS, Ioannis; STAMATATOS, Efstathios. Words versus Character n-Grams for Anti-Spam Filtering. *International Journal on Artificial Intelligence Tools*. 2007-12, roč. 16, s. 1047–1067. Dostupné z DOI: 10.1142/S0218213007003692.
9. LIU, B. Sentiment analysis and subjectivity. *Handbook of Natural Language Processing, Second Edition*. 2010-01, s. 627–666.
10. LIU, Bing. *Sentiment Analysis: Mining Opinions, Sentiments, and Emotions*. Cambridge University Press, 2015. Dostupné z DOI: 10.1017/CBO9781139084789.
11. MEDHAT, Walaa; HASSAN, Ahmed; KORASHY, Hoda. Sentiment analysis algorithms and applications: A survey. *Ain Shams Engineering Journal*. 2014, roč. 5, č. 4, s. 1093–1113. ISSN 2090-4479. Dostupné z DOI: https://doi.org/10.1016/j.asej.2014.04.011.
12. HOWARD, Jeremy; RUDER, Sebastian. *Universal Language Model Fine-tuning for Text Classification*. 2018. Dostupné z arXiv: 1801.06146 [cs.CL].
13. DEVLIN, Jacob; CHANG, Ming-Wei; LEE, Kenton; TOUTANOVA, Kristina. *BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding*. 2018. Dostupné z arXiv: 1810.04805 [cs.CL].

14. YANG, Zhilin; DAI, Zihang; YANG, Yiming; CARBONELL, Jaime; SALAKHUTDINOV, Ruslan; LE, Quoc V. *XLNet: Generalized Autoregressive Pretraining for Language Understanding*. 2019. Dostupné z arXiv: 1906.08237 [cs.CL].

15. SHALEV-SHWARTZ, Shai; ZHAN, Tong. Stochastic dual coordinate ascent methods for regularized loss minimization. *Journal of Machine Learning Research 14.Feb*. 2013, s. 567–599.

16. TRAN, Kenneth; HOSSEINI, Saghar; XIAO, Lin; FINLEY, Thomas; BILENKO, Mikhail. Scaling Up Stochastic Dual Coordinate Ascent. In: *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. 2015, s. 1185–1194. KDD '15. ISBN 9781450336642. Dostupné z DOI: 10.1145/2783258.2783412.

17. JOULIN, Armand; GRAVE, Edouard; BOJANOWSKI, Piotr; MIKOLOV, Tomas. *Bag of Tricks for Efficient Text Classification*. 2016. Dostupné z arXiv: 1607.01759 [cs.CL].

18. MIKOLOV, Tomas; CHEN, Kai; CORRADO, Greg; DEAN, Jeffrey. *Efficient Estimation of Word Representations in Vector Space*. 2013. Dostupné z arXiv: 1301.3781 [cs.CL].

19. GOLDBERG, Yoav. Neural Network Methods for Natural Language Processing. *Synthesis Lectures on Human Language Technologies*. 2017-04, roč. 10, s. 1–309. Dostupné z DOI: 10.2200/S00762ED1V01Y201703HLT037.

20. GOODFELLOW, Ian; BENGIO, Yoshua; COURVILLE, Aaron. *Deep Learning*. MIT Press, 2016. http://www.deeplearningbook.org.

21. KIM, Yoon. *Convolutional Neural Networks for Sentence Classification*. 2014. Dostupné z arXiv: 1408.5882 [cs.CL].

22. PHI, Michael. *Illustrated Guide to LSTM's and GRU's* [online]. 2018-09 [cit. 2020-04-24]. Dostupné z: https://towardsdatascience.com/illustrated-guide-to-lstms-and-gru-s-a-step-by-step-explanation-44e9eb85bf21.

23. KOTZIAS, Dimitrios; DENIL, Misha; FREITAS, Nando de; SMYTH, Padhraic. From Group to Individual Labels Using Deep Features. In: *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. 2015, s. 597–606. KDD '15. ISBN 9781450336642. Dostupné z DOI: 10.1145/2783258.2783380.

24. MAAS, Andrew L.; DALY, Raymond E.; PHAM, Peter T.; HUANG, Dan; NG, Andrew Y.; POTTS, Christopher. Learning Word Vectors for Sentiment Analysis. In: *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*. 2011-06, s. 142–150.

25. ZHANG, Xiang; ZHAO, Junbo; LECUN, Yann. Character-Level Convolutional Networks for Text Classification. In: *Proceedings of the 28th International Conference on Neural Information Processing Systems - Volume 1*. MIT Press, 2015, s. 649–657. NIPS'15.

26. MIKOLOV, Tomas; SUTSKEVER, Ilya; CHEN, Kai; CORRADO, Greg; DEAN, Jeffrey. *Distributed Representations of Words and Phrases and their Compositionality*. 2013. Dostupné z arXiv: `1310.4546 [cs.CL]`.

A Kompletní výsledky předzpracování textu

	SDCA	Averaged Perceptron	FastText	LSTM	CNN
základní předzpracování	82,24	75,95	58,30	80,80	80,50
odstranění interpunkce	81,22	77,30	59,90	80,30	81,10
lemmatizace	80,79	76,48	57,10	81,50	82,40
odstranění stop slov	78,55	73,84	61,80	76,80	77,60
odstranění stop slov + odstranění interpunkce	77,24	75,13	68,70	75,80	77,80
odstranění stop slov + lemmatizace	75,72	73,24	65,20	76,80	78,10

Tabulka 26: Výsledky všech technik pro předzpracování textu pro všechny klasifikační metody na datové sadě Yelp (v procentech)

	SDCA	Averaged Perceptron	FastText	LSTM	CNN
základní předzpracování	86,57	87,69	89,23	89,43	88,90
odstranění interpunkce	86,60	87,77	88,94	89,47	88,67
lemmatizace	86,00	87,32	88,92	89,41	89,02
odstranění stop slov	85,29	86,36	88,68	89,74	89,80
odstranění stop slov + odstranění interpunkce	85,27	86,50	88,46	89,49	89,25
odstranění stop slov + lemmatizace	84,56	86,04	87,90	89,70	89,72

Tabulka 27: Výsledky všech technik pro předzpracování textu pro všechny klasifikační metody na datové sadě IMDB (v procentech)

	SDCA	Averaged Perceptron	FastText	LSTM	CNN
základní předzpracování	87,86	90,63	90,85	94,12	93,07
odstranění interpunkce	88,45	90,69	90,68	93,80	92,83
lemmatizace	87,93	90,28	90,30	93,84	92,27
odstranění stop slov	87,28	89,90	89,96	91,67	90,77
odstranění stop slov + odstranění interpunkce	87,47	89,50	89,80	91,44	90,63
odstranění stop slov + lemmatizace	86,95	89,17	89,56	91,30	90,61

Tabulka 28: Výsledky všech technik pro předzpracování textu pro všechny klasifikační metody na datové sadě Amazon (v procentech)

B Elektronická příloha

Obsahem elektronické přílohy je:

- složka *data*:
  - Datové sady Yelp, IMDB a jejich lemmatizované verze.
  - Python skript *data_downloader*, jenž stáhne a extrahuje největší datovou sadu Amazon a její lemmatizovanou verzi.
- složka *FastText*:
  - Python skripty a Jupyter notebooky pro metodu FastText, rozdělené podle experimentů a datových sad
- složka *ML_NET*:
  - Kompletní adresář s projektem *Visual Studio 2017*, obsahuje všechny experimenty na všech datových sadách s metodami SDCA a Averaged perceptron.
- složka *neural_networks*:
  - Python skripty a Jupyter notebooky pro obě neuronové sítě, rozdělené podle experimentů a datových sad
- složka *lemmatization*:
  - Projekt *Visual Studio 2017*, určen pro lemmatizaci datových sad.

Každá složka obsahuje textový soubor *README*, který obsahuje další informace o obsahu dané složky, pokyny pro spuštění a seznamy potřebných knihoven.